UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN
BOOKSTACKS

Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

http://www.archive.org/details/minimumtardiness153rama

FACULTY WORKING PAPER 93-0153

330
B385
1993:153 COPY 2
STX

THE LIBRARY OF THE
SEP 22 1993
UNIVERSITY OF ILLINOIS
URBANA-CHAMPAIGN

# Minimum Tardiness Scheduling in Flow Shops: Construction and Evaluation of Alternative Solution Approaches

*Narayan Raman*
*Department of Business Administration*
*University of Illinois*

Bureau of Economic and Business Research
College of Commerce and Business Administration
University of Illinois at Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 93-0153

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

August 1993

# Minimum Tardiness Scheduling in Flow Shops: Construction and Evaluation of Alternative Solution Approaches

Narayan Raman
Department of Business Administration

# MINIMUM TARDINESS SCHEDULING IN FLOW SHOPS: CONSTRUCTION AND EVALUATION OF ALTERNATIVE SOLUTION APPROACHES

Narayan Raman

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, Illinois 61820

August 1993

# EXECUTIVE SUMMARY

Surveys of industrial scheduling practice show that meeting customer due dates is a critical concern for many manufacturing systems. While there is considerable research on the effectiveness of scheduling rules in job shops, very little work is reported on flow shops. Although the scheduling rules developed for job shops can be applied in flow shops as well, we show that the inherent structure of a flow shop can be utilized to construct an effective solution procedure. In particular, for the total tardiness problem, we develop conditions for local optimality in a 2-machine flow shop, and use these results to generate an efficient improvement heuristic procedure. We also construct solution methods that utilize the notion of shifting bottlenecks.

Our computational experience reveals the superiority and robustness of the proposed approaches under a variety of problem scenarios. The results of this study indicate the need to look beyond permutation schedules for solving the flow shop tardiness problem. Furthermore, they suggest that the criterion for determining machine criticality needs to take the scheduling objective into consideration. While it appears intuitive to impute criticality on the basis of machine workloads, and thereby, consider the machines for scheduling in the order of their workloads, this approach may frequently result in inferior shop performance.

## 1. INTRODUCTION

Surveys of industrial scheduling practice show that meeting customer due dates is a critical concern for many manufacturing systems (Panwalker et al. 1973, Smith et al. 1986). This has generated substantive research on due date based scheduling objectives such as minimizing total job tardiness. Much of this research deals with the relative effectiveness of priority dispatching rules (see, for example, Conway 1965, Carroll 1965, Baker and Bertrand 1982, Kanet and Hayya 1982, Baker and Kanet 1983, Baker 1984 and Vepsalainen and Morton 1987) under different problem scenarios in *dynamic* job shops. A parallel body of work investigates dominance conditions that insure local optimality between adjacent jobs in any schedule for *static* single machine problems (see, for example, Rinnooy Kan 1976).

However, very little work is reported on the static tardiness problem in flow shops, although they represent many real systems. While the scheduling rules developed for job shops can clearly be applied in flow shops, it may be possible to generate more effective schedules by exploiting their inherent structure as done by Ow (1985) and Morton and Pentico (1993). Following these studies, flow shop scheduling approaches can be classified as either *centralized* or *decentralized*. A centralized approach generates the schedule for the entire flow shop simultaneously and in order to do so, it utilizes global information. However, the resulting scheduling problem is quite difficult to solve. On the other hand, the decentralized approach schedules only one machine at a time. While it strives only for local optimality, the resulting

problem is usually easier to solve (also see Ow 1985 for a related discussion).

Morton and Pentico (1993) describe several scheduling procedures under these two classes. Ow's (1985) procedure provides a compromise between the centralized and decentralized approaches. She develops conditions for local optimality between adjacent jobs in a 2-machine flow shop. She utilizes these conditions for developing a heuristic solution method that focuses on the bottleneck machine in a proportionate flow shop. However, its performance in a general flow shop is unknown. It is useful to also note that Ow's approach results in a *permutation* schedule; while these schedules are generally believed to be effective, it may be possible to generate nonpermutation schedules that are superior.

This paper has a two objectives. First, we construct two alternative decentralized scheduling approaches to the general flow shop tardiness problem. The first approach is based on developing alternative conditions of local optimality for a 2-machine system. However, unlike Ow (1985), these conditions are based on considering only *one machine at a time*. They also provide theoretical insight into the merit of focusing on the bottleneck as done by Ow. We use these conditions to construct an *active* nonpermutation schedule for the general flow shop problem by decomposing it into several single machine problems. The second approach utilizes the notion of *shifting bottlenecks* originally proposed by Adams, Balas and Zawack (1988) for solving the job shop makespan problem. We propose alternative ways in

which this approach can be adapted for minimizing tardiness in a flow shop.

Second, we present a comprehensive evaluation of several centralized and decentralized scheduling methods. These include methods developed in this study as well as others proposed in previous research. While minimizing mean tardiness (MT) is the primary criterion of schedule effectiveness, we also investigate the robustness of each procedure by evaluating it for the measures of proportion of tardy jobs (PT) as well as mean flow time (MFT). While PT is clearly a due date based scheduling measure, we believe that the ability of any system to quote and maintain short job due dates is closely linked with its ability to control MFT.

The paper is organized as follows. The minimum tardiness problem is formulated in § 2. We also present a network representation of this problem in order to better describe scheduling approaches that are presented in the later sections. We develop conditions for local optimality for 2-machine flow shops in § 3. In § 4 and § 5, respectively, we describe the various centralized and decentralized scheduling approaches proposed in past research. In § 5, we additionally construct a scheduling method that utilizes the results developed in § 3. We also present the alternative scheduling approach that is based on shifting bottlenecks. Our computational experience is discussed in § 6, and the main results of this paper are summarized in § 7.

## 2. THE FLOW SHOP TARDINESS PROBLEM

We consider a static problem with N jobs that are all available for scheduling at time zero in an M-machine flow shop. Let $J = \{1,\ldots,N\}$ denote the set of these jobs. The input parameters include the operation processing times $p_{jm}$ of job j on machine m, job processing times $p_j = \sum_{m=1}^{M} p_{jm}$, and job due dates $d_j$. The primary decision variable is $c_{jm}$--the time at which job j finishes processing on machine m. Let $c_j = c_{jM}$ be the completion time, $T_j = \max(0, c_{jM} - d_j)$ be the tardiness, and $E_j = \max(0, d_j - c_{jM})$ be the earliness of job j. The minimum tardiness problem (MTP) is stated as

**MTP**

$$\max \sum_{j=1}^{N} T_j \tag{1}$$

$$c_{jm} - c_{im} \geq p_{jm} \vee c_{im} - c_{jm} \geq p_{im}, \quad \forall\, j,i \in J, \text{ and } \forall m \tag{2}$$

$$c_{j1} \geq p_{j1}, \quad \forall j \tag{3}$$

$$c_{jm} - c_{j,m-1} \geq p_{jm}, \quad m = 2,\ldots,M \tag{4}$$

$$c_{jM} + E_j - T_j = d_j, \quad \forall j \tag{5}$$

$$c_{jm} \geq 0, \ \forall j,m; \ E_j, T_j \geq 0, \ \forall j \tag{6}$$

(1) specifies the objective of minimizing total tardiness. (2) insure that no more than one operation is processed on a machine at any time. (3) and (4), respectively, require that the first operation of any job cannot start before the job is released, and any subsequent operation is scheduled only after its predecessor is completed. (5) define job tardiness while (6) specify the nature of the variables.

Following Balas (1970), and Adams, Balas and Zawack (1988), **MTP** can be represented as a network as shown in Figure 1. The nodes in this graph, other than the dummy start and finish nodes, depict individual

operations, while the directed *conjunctive* arcs, shown in solid lines, indicate the order in which the operations in each job are to be processed. The disjunctions formed by constraints (2) are shown by the broken arcs with double-ended arrows; these *disjunctive* arcs relate to the sequence in which the jobs are processed at each machine. In this network representation, a disjunctive arc is said to be *settled* whenever its direction is determined by assigning a relative order of processing between the two operations linked by that arc. Finding a complete schedule is equivalent to settling all pairs of disjunctive arcs.

---

Insert Figure 1 Here

---

**MTP** is a difficult problem to solve optimally even for single machine systems (Du and Leung 1990). However, limited results are available for a 2-machine flow shop. First, a permutation schedule is optimal in these systems. Ow (1985) also develops conditions under which a schedule is locally optimal, i.e., two adjacent jobs follow a given order. In the next section, we develop alternative conditions of local optimality.

## 3. THE 2-MACHINE FLOW SHOP

Consider an arbitrary schedule in a 2-machine flow shop in which jobs i and j are processed consecutively. Let $t_1$ and $t_2$ be the times at which machines 1 and 2 become available for processing i and j. Figure 2 depicts two of the several configurations that are possible for the case in which i precedes j (denoted by $i \prec j$). [Note that we need consider only permutation schedules since they are optimal in a 2-machine flow shop.]

---

Insert Figures 2a and 2b Here

---

Let $x^+ = \max(x,0)$ and $x^- = \min(x,0)$; then, as shown in Figure 2a, $S_i^+ = (0, t_1+p_{im}-t_2)^+$ is the *leading* idle time resulting on machine 2 when $i \prec j$. In the alternative scenario depicted in Figure 2b, $S_i^- = (0, t_1+p_{im}-t_2)^-$ is the waiting time of i at machine 2, while $(S_i^-+p_{j1}-p_{i2})^+$ is the *trailing* idle time on this machine. [The leading idle time $S_j^+$ and the trailing idle time $(S_j^-+p_{i1}-p_{j2})^+$ when $j \prec i$ are similarly defined.] Note that, in any schedule, $S_i^+ S_i^- = 0$. Let $c_i^i$ ($c_j^i$) denote the completion time of i (j) on machine 2 when $i \prec j$. Then

$$c_i^i = t_2 + S_i^+ + p_{i2}, \text{ and } c_j^i = t_2 + p_{i2} + S_i^+ + (S_i^- + p_{j1} - p_{i2})^+ + p_{j2}.$$

Ow shows that a schedule in which $i \prec j$ is locally optimal only if

$$Pr_{ij}(t_1,t_2) > Pr_{ji}(t_1,t_2), \tag{7}$$

where

$$Pr_{ij}(t_1,t_2) = \frac{1}{p_{i2} + S_i^+}\left[1 - \frac{(d_i - c_i^i)^+ + S_i^+ - (S_j^- + p_{i1} - p_{j2})^+}{p_{j2} + S_j^+}\right]^+$$

is the priority of i relative to j; $Pr_{ji}(t_1,t_2)$ is similarly defined. Note that this result requires considering both machines for computing priority values. We now develop an alternative set of conditions that relate to individual machines. Let $r_{im}$ be the ready time of job i on machine m in any partial schedule; then $r_{i1} = 0$, and $r_{i2} = c_{i1} = t_1 + p_{i1}$.

Define the operation due date $d_{im}$ of o on machine m as follows; $d_{i2} = d_i$; $d_{i1} = d_i - p_{i2}$, and the priority of job i at machine m as

$$P_{im} = \max\{\max(r_{im}, t_m) + p_{im}, d_{im}\} + \max(r_{im}, t_m). \quad (8)$$

Next, we introduce the notion of *schedule conflict*. Job j is said to conflict with job i on machine m in any schedule, denoted by $i \overset{m}{\Longleftarrow} j$, if j succeeds i and is available for processing on machine m before i is completed, i.e., $r_{jm} < c_{im}$. Note that in this case, the trailing idle time $(S_i^- + p_{j1} - p_{i2})^+$ is zero. The situation in which i conflicts with job j, denoted by $j \overset{m}{\Longleftarrow} i$, is similarly defined. Clearly, both i and j conflict with each other on machine 1, i.e., $i \overset{1}{\Longleftrightarrow} j$. However, four configurations are possible on machine 2:

1. There is no conflict, denoted by $j \overset{2}{\not\Longleftrightarrow} i$, as shown in Figure 3,
2. Both jobs conflict with each other, or $i \overset{2}{\Longleftrightarrow} j$, as shown in Figure 4,
3. j conflicts with i, $i \overset{2}{\Longleftarrow} j$, $j \overset{2}{\not\Longleftarrow} i$, as shown in Figure 5, and
4. i conflicts with j, $j \overset{2}{\Longleftarrow} i$, $i \overset{2}{\not\Longleftarrow} j$, as shown in Figure 6.

---

Insert Figures 3 through 5 Here

---

[Note that Figures 3 through 6 show only one instance of each configuration; in particular, in all these instances, $S_i^+ = S_j^+ = 0$.] We now state the conditions for local optimality.

**Remark 1.** If $i \overset{2}{\Longleftrightarrow} j$, then $i \prec j$ in an optimal solution if and only if $P_{i2} \leq P_{j2}$.

**Proof:** Refer to Appendix 1.

**Remark 2.** If $i \overset{2}{\not\Longleftrightarrow} j$, then $i \prec j$ in an optimal solution if and only if $P_{i1} \leq P_{j1}$.

**Proof:** Refer to Appendix 2.

**Remark 3.** Let $j \overset{2}{\Longleftarrow} i$, but $i \overset{2}{\not\Longleftarrow} j$. If $P_{i1} \leq P_{j1}$, then $P_{i2} \leq P_{j2}$, and i precedes j in an optimal schedule. Alternatively, if $P_{j2} \leq P_{i2}$, then $P_{j1} \leq P_{i1}$, and j precedes i in an optimal solution.

**Proof:** Refer to Appendix 3.

Remarks 1 and 2 provide some insight into the merit of focusing on the bottleneck machine in order to compute relative job priorities as proposed by Ow (1985) and Morton and Pentico (1993). Note that schedule conflicts are more likely to occur on bottleneck machines. Also recall that i and j always conflict on machine 1. If they do not conflict on machine 2, as in configuration 1, then Remark 1 indicates that machine 1 is the critical machine, and the overall relative priorities of i and j should be based on $P_{i1}$ and $P_{j1}$. Alternatively, if conflicts do occur on machine 2, as shown in configuration 2, then $P_{i2}$ and $P_{j2}$ computed on this machine should determine the overall relative priority of these two jobs. This remark also indicates that when conflicts occur at both machines, the second machine is dominant.

Remark 3 indicates that, in the case of one-way conflicts as in configurations 3 and 4, the relative order of priorities $P_{im}$ and $P_{jm}$ remains the same on both machines. Selecting the job with the smaller P is sufficient for local optimality.

## 4. CENTRALIZED SCHEDULES

We now describe the various solution approaches proposed in the literature for the flow shop mean tardiness problem. Wherever necessary, we also discuss the modifications made in these procedures for the purpose of this study. In this section, we consider only centralized procedures; the decentralized methods are discussed in § 5.

As mentioned earlier, Ow (1985) suggests a procedure that focuses on the known bottleneck in the system. The system is then treated as a 2-machine flow shop comprising the first machine and the bottleneck machine, and the local optimality conditions (7) are used to determine the relative ordering of individual jobs. Ow proposes an alternative form of the priority function that computes the priority of job i as

$$Pr_i = \frac{1}{p_{ib} + S_i^*} \exp\left[ -\frac{(d_i - c_i^i)^+ + h(S_i^* - S)}{k(\bar{p} + S)} \right] \qquad (9)$$

where b is the bottleneck machine, h is a measure of the opportunity cost of a time unit on b, k is a look-ahead parameter, S is the average idle time on b, and p is the average processing time on b. The job with the maximum priority is processed next. In terms of the results of § 3, this approach essentially resolves the scheduling conflicts on the bottleneck machine. In so doing, it ignores conflicts that arise at

machines downstream to the bottleneck machine. As Ow notes, it is reasonable to assume in a proportionate flow shop that the order in which jobs are completed at the last machine is the same as the order on the bottleneck machine.

In order to apply this rule in a general flow shop, we modify it as follows. Each machine in the flow shop is considered to be a bottleneck in turn, and a schedule for the entire shop is developed in the manner described by Ow. Among the M schedules generated, we select one with the minimum total tardiness. Henceforth, we call this approach the *Modified Focused Scheduling* (MFS) method. Consistent with Ow (1985), this approach develops a permutation schedule for the shop.

Morton and Pentico (1993) describe *Botflow* family of rules that generate the schedule for the entire flow shop by solving the single machine tardiness problem with respect to job ready times and operation due dates at the bottleneck machine. These problems are solved either in a single pass or through multiple iterations. The sequence of jobs obtained eventually is enforced on each machine in the system to obtain a permutation schedule.

In this study, we implement the *Botflow 3* method which is the most general among the Botflow rules. Similar to MFS, this method develops a schedule based on each machine in turn, and eventually selects the best among them. The job ready time and operation due date values used in the single machine problem are updated iteratively. Suppose that the machine under consideration is m. Let $r_{im}(k)$, $d_{im}(k)$, and $c_{im}(k)$ denote, respectively, the ready time, operation due date and completion time of

job i at machine m in iteration k. Then the job ready times at the beginning of the first iteration are determined by

$$r_{im}(1) = \begin{cases} 0, & m = 1 \\ \sum_{\ell=1}^{m-1} p_{i\ell}, & m = 2,\ldots,M \end{cases} \tag{10}$$

and the initial operation due dates by

$$d_{im}(1) = \begin{cases} ap_{i1}, & m = 1 \\ d_{i,m-1} + ap_{im}, & m = 2,\ldots,M \end{cases} \tag{11}$$

for i = 1,…,N, where $a = d_i/p_i$ is the flow allowance factor. Subject to these parameters, jobs are sequenced according to the Rachamadugu and Morton (Rachamadugu and Morton 1982) heuristic. According to this method, the priority of job i on machine m is given by

$$A_{im} = \frac{1}{p_{im}} \exp\left\{-\left[\frac{d_{im} - (t_m + p_{im})}{k\bar{p}_m}\right]^+\right\}$$

where $\bar{p}_m = \frac{1}{N}\sum_{i=1}^{N} p_{im}$ is the average job processing time on machine m, k is a look-ahead parameter, and $t_m$ is the time at which machine m is next available. At this time, only those jobs that are available at the machine are considered, and the job with the maximum $A_{im}$ is selected. The relative job order obtained in this manner is implemented at all machines to obtain the complete shop schedule. At the next iteration, $r_{im}$ reflects the total processing and waiting time incurred by i prior to its arrival at m in the previous iteration. $d_{im}$ is similarly

modified to reflect the actual lead time incurred by i after it is processed at m at the previous iteration. In particular, at the $k^{th}$ iteration,

$$r_{im}(k) = \begin{cases} 0, & m = 1 \\ c_{i,m-1}(k-1), & m = 2,\ldots,M \end{cases} \qquad (13)$$

and

$$d_{im}(k) = d_i - [c_{iM}(k-1) - c_{im}(k-1)]. \qquad (13)$$

Iterations stop when there is no improvement in the tardiness values obtained in two consecutive iterations. This approach is implemented for each machine in turn, and the best among the M shop schedules generated in this manner is selected. Henceforth, we call this rule Botflow.

## 5. DECENTRALIZED SCHEDULES

Decentralized schedules can be further categorized into *constructive* schedules and the shifting bottleneck schedules. These two categories are now described.

### 5.1 Constructive Schedules

Constructive schedules are developed chronologically. They deal with individual machines in the order they become available for processing the waiting jobs. Because of the fixed sequence in which these machines are visited by all jobs, they can be scheduled in the order of their indexes. In terms of the network shown in Figure 1, this approach starts with the left and proceeds to the right. At a given

machine, the job to be taken up for imminent processing is selected by a priority dispatching rule. Morton and Pentico (1993) evaluate a number of different priority rules, such as First-come-first-serve, Weighted Shortest Processing Time, Earliest Due Date, Slack per Remaining Operation, COVERT, RM-1 and RM-Iter. They report the superiority of RM-1 and RM-Iter rules based on their computational experience; consequently, in this paper, we investigate these two rules only.

RM-1 implements the Rachamadugu and Morton (1982) heuristic for solving the single machine tardiness problem at each machine. This approach constructs a nondelay schedule by considering machines 1 through M in that order in a single pass. At each machine, the job ready times and due dates are computed by using (10) and (11). However, unlike Botflow, RM-1 generates nonpermutation schedules because priority dispatching is done at each machine individually.

RM-Iter implements RM-1 while updating $r_{im}$ and $d_{im}$ in an iterative manner according to (12) and (13). The procedure terminates when no improvement in tardiness values is observed in consecutive iterations.

The third constructive approach considered in this study is based on extending the results in Remarks 1 through 3 to yield an iterative improvement heuristic in a general flow shop. This procedure starts with a feasible *nondelay* schedule, and iteratively improves upon it by resequencing jobs to eventually generate an *active* schedule.

At any iteration, the machines are scanned in the order of their index, and at each machine, the schedule is scanned from front to back. Suppose that j and i are two adjacent jobs on any machine m, such that $j \prec i$ in the given schedule, and these two jobs conflict, i.e.,

$r_{im} < c_{jm}$. Then i and j are candidates for a switch if $P_{im} \leq P_{jm}$, where $P_{im}$ and $P_{jm}$ are obtained from (8). The switch occurs if

$$\delta T(i,j,m) = \sum_{k \in S_{ij}^{m}} (T'_{km} - T_{km}) < 0$$

where $S_{ij}^{m}$ is the set of jobs comprising i, j and all jobs that follow these two jobs, and $T_{km}$ and $T'_{km}$, respectively, are the tardiness of k before and after the switch. Note that (8) computes priorities even for those jobs that are currently not on the machine by considering their ready times. Whenever such a job is schedule ahead of another job that is currently available at the machine, forced idle time results on the machine. This situation is depicted in Figure 7 in which job i has a ready time $r_{im} > t_m \geq r_{jm}$.

---

Insert Figure 7 Here

---

If $P_{im} \leq P_{jm}$, then it is locally optimal to schedule i ahead of j. However, the resulting forced idle time at this machine may delay the completion of one or more jobs following i and j which, in turn, could result in higher tardiness for these jobs. Let $I_m = (r_{im} - t_m)^+$ be the forced idle time induced by scheduling i ahead of j. Then, denoting the completion time of the job in position k by $c_{[k]m}$ and $c'_{[k]m}$ before and after the switch, respectively, and assuming that the order of jobs following i and j do not change because of the switch, we have

$$c'_{[k]m} = \max(c_{[k]m}, c'_{[k-1]m} + I_m)$$

and

$$T'_{[k]m} = \max(0, c'_{[k]m} - d_{[k]m})$$

for all jobs that follow i and j. Note that, while computing the revised completion times, if we find a job for which there is no change in the completion time, i.e., a job in position k such that $c_{[k]m} = c'_{[k]m}$, we need not consider any of the remaining jobs in the schedule since their completion times and tardiness values will remain unchanged after the switch.

The initial schedule is a nondelay schedule that is generated by using the RM-1 method. Similar to RM-Iter, job ready times and operation due dates in subsequent iterations are determined by (12) and (13); and the procedure terminates when there is no improvement in the tardiness values obtained in two consecutive iterations. Henceforth, we refer to this solution method as the *Flow Shop Decomposition* (FSD) procedure.

**5.2 The Shifting Bottleneck Schedules**

Shifting bottleneck (SB) schedules are similar to constructive schedules in that they deal with only one machine at a time. However, while constructive schedules consider machines in the order of their indexes, SB schedules consider them in the order of their criticality. As each machine is fully scheduled, the criticality indexes of the remaining machines are updated. When all machines are scheduled at the end of a cycle, this process is repeated for the next cycle; the

procedure is terminated when no improvement in the solution value occurs for given number of cycles. The shifting bottleneck approach is used successfully by Adams et al. (1988) for the job shop makespan problem.

Under the SB approach, the problem to be solved at a given machine requires minimizing total tardiness subject to the operation arrival times and operation due dates obtained by considering only the *settled* arcs. Initially, these comprise only the conjunctive arcs representing the precedence relationships within each job; the ready times and operation due dates are determined by (10) and (11). The resulting solution assigns an order on the processing of the various jobs on this machine, and thereby, settles the disjunctive arcs corresponding to that machine. In the general case, determining the ready times and operation due dates requires solving two longest path problems with respect to the settled arcs in the network given in Figure 1 (Adams et al. 1988).

It is possible to construct alternative procedures within the overall framework of the shifting bottleneck approach. These procedures will differ in how the criticality of a machine is defined, and the manner in which the single machine tardiness problem is solved. In this study, we use two alternative ways of defining machine criticality. In the first method, it is determined by the relative machine workload. Under this approach, machines are scheduled in the order of their workloads. In the second approach, machine criticality is measured by the magnitude of total tardiness obtained by solving the individual single machine problems. In particular, we solve the single machine tardiness problem with respect to the settled arcs on each unscheduled machine at the beginning, and thereafter whenever a machine is

scheduled. The machine at which the maximum value of total tardiness is realized is next selected for consideration. In the following, we refer to these two approaches as SB1 and SB2, respectively.

The single machine tardiness problem is solved by applying the Rachamadugu and Morton (1982) heuristic to yield a nondelay schedule. Under each of SB1 and SB2 approaches, we additionally generate an active schedule in a manner similar to FSD.

## 6. COMPUTATIONAL STUDY

The experimental study evaluates the relative performance of the various centralized and decentralized scheduling procedures described in § 4 and § 5. In particular, we consider MFS and Botflow among the centralized, and RM-1, RM-Iter, FSD, SB1 and SB2 among the decentralized methods. Both nondelay and active versions of SB1 and SB2 were implemented; however, in order to simplify presentation, we report the results of only the active schedules since they were superior in most cases.

Consistent with Ow (1985), we use $h = 20$, and $k = 2.0$ in (9) while implementing MFS. Similarly, following Vepsalainen and Morton (1988), we set $k = 2.0$ in RM-1 and RM-Iter. As proposed in Adams et al. (1988), we incorporate local reoptimization in both SB1 and SB2 whenever a machine is schedule. This step essentially resequences each of the scheduled machines with respect to the most recent set of settled arcs. Reoptimization is carried up to three iterations; however, when all machines are scheduled, reoptimization continues until no further improvement is observed.

### 6.1 Experiment Design

Two parameters--Z and R are used to control the tightness and the variation of job due dates respectively. The tardiness factor Z is an approximate measure of the proportion of tardy jobs, while R determines the range of job due dates. For given Z and R, job due dates are sampled from a uniform distribution in the interval $[\bar{d}(1 - R/2), \bar{d}(1 + R/2)]$, where $\bar{d} = c_{max}(1 - Z)$ is the average job due date, and $c_{max}$ is the makespan of the sequence obtained by scheduling all operations on a first-come-first-serve basis at each machine. Z and R have been used extensively for generating test data in single machine tardiness problems (see, for example, Vepsalainen and Morton 1988). Because of the forced machine idle times, Z is only an approximate measure of the proportion of tardy jobs in a job shop. Nonetheless, it helps anchor due date tightness at various levels.

Individual problem scenarios are generated by varying one or more of the parameters: i) *Number of machines M*, this was considered at two levels--4 and 8, ii) *number of jobs N*, this was considered at two levels--25 and 50, iii) Z, this was considered at three levels--0.25, 0.50 and 0.75, and iv) R, this was considered at two levels--0.50 and 1.50. Within each scenario, ten problems are randomly generated by sampling operation processing times from a uniform distribution in the interval [1, 100]. The average solution value across these instances under each scheduling approach is presented in Tables 1 through 6. We also report in parentheses the number of times a given scheduling rule is found to be the best among those tested across these ten instances. In total, 240 problems are solved for each scheduling approach.

The primary performance measure is total tardiness (TT). For reporting purposes, we use the normalized value given by $\sum_j T_j / \sum_j p_j$. In order to evaluate the robustness of each method, we also report the proportion of tardy jobs (PT) and the total job flow time (TFT) values. Similar to total tardiness, we normalize total flow time with respect to the sum of job processing times.

**6.2 Experimental Results**

Tables 1 and 2 give the total tardiness values for M = 4 and M = 8, respectively. These results show that, at low Z values, SB1 performs the best when the due date range is small, while Botflow, SB2 and RM-Iter are the best rules for larger R values. This results holds across all tested values of M and N. However, for medium and high values of Z, FSD is clearly the best rule in terms of the tardiness values as well as the number of times it finds the best solution. The performance of other rules is somewhat mixed with SB1, SB2, RM-Iter and Botflow doing well under different scenarios. Overall, FSD performs the best, followed by RM-Iter, SB2 and Botflow. While an increase in Z leads to improved performance of MFS and RM-1, it results in a rapid deterioration in the performance of SB1.

---
Insert Tables 1 and 2 Here
---

For the measure of PT, Tables 3 and 4 indicate that MFS and FSD are the best methods overall with MFS somewhat better for smaller, 4-machine system while FSD does better for the larger system. In general, these two rules yield comparable values. Among the other rules Botflow is effective at low Z values. The relative performance of RM-1 and RM-Iter

improves with an increase in Z, while SB1 and SB2 progressively perform poorly; this is particularly so for SB1.

---

Insert Tables 3 and 4 Here

---

Tables 5 and 6 report the TFT values for the various rules. For the 4-machine, 25-jobs system, RM-1 and MFS return the best values when Z is small, while FSD becomes increasingly superior as Z increases. It remains superior for all Z and R values when N = 50, and also for the 8-machine, 25-job system. For the 8-machine, 50-job system, FSD continues to be effective; in this case, MFS, SB2 and RM-Iter also perform well. Overall, FSD is seen to be the best rule; MFS is the next best, followed by RM-1 and RM-Iter. All these rules are also generally robust. Similar to the other two criteria, the performance of SB1 deteriorates with an increase in Z.

---

Insert Tables 5 and 6 Here

---

In summary, FSD has the best overall performance among the various rules tested. It is better, in particular, for medium and large values of Z; however, it remains robust across all scenarios for all three performance measures tested. Among the other methods, RM-Iter and, to a lesser extent, SB2 and RM-1 perform well in terms of both effectiveness and robustness. MFS exhibits shortest processing time rule (SPT) like properties (Baker 1984). It yields low PT and TFT values, and also does well for the TT criterion when tardiness levels are high. Botflow does well for small systems, low tardiness factor and high due date ranges. Between the two shifting bottleneck approaches, SB2 is generally more

effective for all three measures. SB1 is quite sensitive to Z, its performance deteriorates rapidly when due dates become increasingly tighter.

## 7. SUMMARY

This paper examines alternative scheduling approaches for minimizing tardiness in a flow shop. In view of the problem complexity, we consider only heuristic solution methods. While job shop scheduling rules can clearly be used in a flow shop as well, we show that the inherent structure of a flow shop can be utilized to construct an effective, decomposition based solution method. In particular, we develop conditions for local optimality in a 2-machine flow shop, and use these results to generate an improvement heuristic procedure. We also construct solution procedures that utilize the notion of shifting bottlenecks.

In an extensive computational study, we compare these methods with others that have been proposed in past research, and show their effectiveness under a variety of problem scenarios and across the scheduling measures of proportion of tardy jobs and total flow time as well. This study provides two other results that should be of interest to both researchers and practitioners. First, the experimental results show that permutation schedules may not perform very well for some scheduling criteria such as total tardiness. While nonpermutation schedules are less restrictive, and therefore, are likely to be more effective, they usually require more computational effort. The results of this study indicate that the difference in the tardiness values may

be large enough to justify incurring this additional computational burden.

Second, the difference in the performance of SB1 and SB2 suggest that the criterion for determining the bottleneck machine needs to problem specific. While it appears reasonable to deem the machine with the largest workload to be the most critical machine as done in SB1, this approach may actually lead to inferior schedules. In this study, SB2, which defined criticality on the basis of tardiness values instead, resulted in superior performance.

**REFERENCES**


1. Adams, J., E. Balas and D. Zawack (1988), "The Shifting Bottleneck Procedure in Job Shop Scheduling," *Management Science*, **34**, 391-401.
2. Baker, K. R. (1984), "Sequencing Rules and Due Date Assignments in a Job Shop," *Management Science*, **30**, 1093-1104.
3. Baker, K. R. and J. M. W. Bertrand (1982), "A Dynamic Priority Rule for Sequencing Against Due Dates," *Journal of Operations Management*, **3**, 37-42.
4. Baker, K. R. and J. J. Kanet (1983), "Job Shop Scheduling with Modified Due Dates," *Journal of Operations Management*, **4**, 11-22.
5. Balas, E. (1969), "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," *Operations Research*, **17**, 941-957.
6. Carroll, D. C. (1965), "Heuristic Sequencing of Single and Multiple Component Jobs," *Ph.D. Dissertation*, MIT, Cambridge, MA.
7. Conway, R. W. (1965), "Priority Dispatching and Job Lateness in a Job Shop," *Journal of Industrial Engineering*, **16**, 123-130.
8. Du, Z. and J. Y.-T. Leung (1990), "Minimizing Total Tardiness on One-Machine is NP-Hard," *Mathematics of Operations Research*, **15**, 483-495.
9. Kanet, J. J. and J. C. Hayya (1982), "Priority Dispatching with Operation Due Dates in a Job Shop," *Journal of Operations Management*, **2**, 155-163.
10. Morton, T. E. and D. W. Pentico (1993), *Heuristic Scheduling Systems*, John Wiley and Sons, New York, NY.

11. Ow, P. S. (1985), "Focused Scheduling in Proportionate Flow Shops," *Management Science*, **31**, 852-869.

12. Panwalker, S. S., R. A. Dudek and M. L. Smith (1973), "Sequencing Research and the Industrial Problem," in *Symposium on the Theory of Scheduling*, edited by S. E. Elmaghraby, Springer-Verlag, Berlin.

13. Rachamadugu, R. V. and T. E. Morton (1982), "Myopic Heuristics for the Single Machine Weighted Tardiness Problem," Working Paper #30-82-83, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA.

14. Raman, N. and F. B. Talbot (1993), "The Job Shop Tardiness Problem: A Decomposition Approach," *European Journal of Operational Research*, **69**, *forthcoming*.

15. Rinnooy Kan, A. H. G. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, Netherlands.

16. Smith, M. L., R. Ramesh, R. A. Dudek and E. L. Blair (1986), "Characteristics of U.S. Flexible Manufacturing Systems--A Survey," in *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, 477-486.

17. Vepsalainen, A. P. J. and T. E. Morton (1988), "Improving Local Priority Rules with Global Lead-Time Estimates: A Simulation Study," *Journal of Manufacturing and Operations Management*, **1**, 102-118.

TABLE 1 - *Total Tardiness*

4-Machine System

| $Z; R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| | | | | $N = 25$ | | | |
| 0.25;0.50 | 0.21 (0) | 0.16 (0) | 0.17 (1) | 0.06 (6) | 0.11 (2) | 0.10 (0) | 0.15 (2) |
| 0.25;1.50 | 0.49 (0) | 0.04 (5) | 0.06 (4) | 0.05 (4) | 0.03 (7) | 0.03 (6) | 0.19 (2) |
| 0.50;0.50 | 1.01 (0) | 1.00 (0) | 0.80 (5) | 0.95 (2) | 1.01 (0) | 1.01 (1) | 0.88 (3) |
| 0.50;1.50 | 1.14 (0) | 0.96 (0) | 0.88 (3) | 1.04 (3) | 0.90 (4) | 0.90 (0) | 0.98 (1) |
| 0.75;0.50 | 2.28 (0) | 2.26 (1) | 1.83 (8) | 3.40 (0) | 2.37 (0) | 2.49 (0) | 2.14 (1) |
| 0.75;1.50 | 2.40 (0) | 2.34 (2) | 2.07 (7) | 3.01 (0) | 2.36 (0) | 2.50 (0) | 2.33 (1) |
| | | | | $N = 50$ | | | |
| 0.25;0.50 | 0.42 (0) | 0.22 (2) | 0.21 (0) | 0.10 (8) | 0.15 (1) | 0.17 (0) | 0.20 (2) |
| 0.25;1.50 | 0.51 (0) | 0.00 (10) | 0.08 (4) | 0.00 (10) | 0.00 (10) | 0.00 (10) | 0.01 (9) |
| 0.50;0.50 | 1.59 (0) | 1.57 (0) | 1.15 (6) | 1.42 (3) | 1.52 (0) | 1.63 (0) | 1.51 (1) |
| 0.50;1.50 | 1.75 (0) | 1.34 (1) | 1.18 (5) | 1.21 (4) | 1.18 (5) | 1.19 (4) | 1.39 (0) |
| 0.75;0.50 | 3.92 (1) | 3.87 (1) | 3.42 (7) | 4.55 (1) | 4.22 (0) | 4.45 (0) | 3.84 (1) |
| 0.75;1.50 | 4.25 (0) | 3.77 (3) | 3.69 (5) | 4.19 (0) | 3.99 (1) | 4.05 (0) | 3.78 (3) |

TABLE 2 - *Total Tardiness*

8-MACHINE SYSTEM

| $Z$; $R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| | | | $N = 25$ | | | | |
| 0.25;0.50 | 0.25 (0) | 0.21 (0) | 0.11 (2) | 0.04 (9) | 0.24 (0) | 0.17 (0) | 0.21 (0) |
| 0.25;1.50 | 0.44 (0) | 0.10 (2) | 0.20 (3) | 0.13 (2) | 0.10 (5) | 0.10 (2) | 0.34 (2) |
| 0.50;0.50 | 0.90 (0) | 0.87 (0) | 0.54 (9) | 1.07 (0) | 0.97 (0) | 0.89 (0) | 0.79 (1) |
| 0.50;1.50 | 0.94 (0) | 0.87 (0) | 0.68 (8) | 1.29 (0) | 0.86 (1) | 0.86 (1) | 0.90 (0) |
| 0.75;0.50 | 1.81 (1) | 1.81 (1) | 1.18 (7) | 2.97 (0) | 1.88 (0) | 1.83 (0) | 1.71 (1) |
| 0.75;1.50 | 1.80 (0) | 1.77 (0) | 1.30 (8) | 2.43 (0) | 1.78 (0) | 1.81 (0) | 1.68 (2) |
| | | | $N = 50$ | | | | |
| 0.25;0.50 | 0.39 (0) | 0.32 (0) | 0.23 (0) | 0.04 (9) | 0.22 (1) | 0.22 (1) | 0.35 (0) |
| 0.25;1.50 | 0.63 (0) | 0.04 (5) | 0.09 (2) | 0.04 (1) | 0.02 (5) | 0.02 (5) | 0.24 (0) |
| 0.50;0.50 | 1.26 (2) | 1.26 (2) | 1.13 (4) | 1.34 (3) | 1.31 (0) | 1.32 (1) | 1.23 (0) |
| 0.50;1.50 | 1.36 (0) | 1.22 (1) | 1.07 (5) | 1.35 (2) | 1.11 (1) | 1.12 (1) | 1.30 (0) |
| 0.75;0.50 | 2.75 (1) | 2.73 (2) | 2.60 (4) | 4.72 (0) | 2.92 (0) | 2.96 (0) | 2.69 (3) |
| 0.75;1.50 | 2.65 (0) | 2.62 (1) | 2.02 (9) | 5.04 (0) | 2.71 (0) | 2.90 (0) | 2.67 (0) |

TABLE 3 - *Proportion of Tardy Jobs*

4-Machine System

| $Z;R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| | | | | $N = 25$ | | | |
| 0.25;0.50 | 0.23 (0) | 0.27 (0) | 0.25 (0) | 0.26 (1) | 0.20 (3) | 0.20 (2) | 0.17 (6) |
| 0.25;1.50 | 0.24 (1) | 0.10 (6) | 0.10 (7) | 0.15 (3) | 0.19 (1) | 0.11 (6) | 0.14 (3) |
| 0.50;0.50 | 0.51 (4) | 0.52 (3) | 0.50 (4) | 0.74 (0) | 0.64 (0) | 0.61 (1) | 0.48 (5) |
| 0.50;1.50 | 0.54 (4) | 0.65 (0) | 0.58 (1) | 0.95 (0) | 0.92 (0) | 0.76 (1) | 0.50 (7) |
| 0.75;0.50 | 0.84 (1) | 0.83 (2) | 0.82 (3) | 0.95 (0) | 0.90 (0) | 0.85 (0) | 0.78 (5) |
| 0.75;1.50 | 0.86 (2) | 0.89 (1) | 0.87 (3) | 0.97 (0) | 0.98 (0) | 0.94 (0) | 0.81 (7) |
| | | | | $N = 50$ | | | |
| 0.25;0.50 | 0.20 (1) | 0.22 (1) | 0.13 (7) | 0.19 (1) | 0.19 (1) | 0.19 (2) | 0.16 (3) |
| 0.25;1.50 | 0.15 (0) | 0.00 (10) | 0.04 (4) | 0.00 (10) | 0.00 (10) | 0.00 (10) | 0.01 (9) |
| 0.50;0.50 | 0.47 (3) | 0.48 (2) | 0.46 (4) | 0.62 (0) | 0.59 (0) | 0.58 (0) | 0.43 (5) |
| 0.50;1.50 | 0.45 (2) | 0.60 (0) | 0.43 (5) | 0.77 (0) | 0.78 (0) | 0.72 (0) | 0.44 (5) |
| 0.75;0.50 | 0.80 (1) | 0.79 (1) | 0.78 (3) | 0.93 (0) | 0.90 (0) | 0.83 (2) | 0.77 (6) |
| 0.75;1.50 | 0.80 (1) | 0.78 (1) | 0.76 (4) | 0.95 (0) | 0.92 (0) | 0.88 (0) | 0.76 (4) |

TABLE 4 - *Proportion of Tardy Jobs*

8-MACHINE SYSTEM

| $Z; R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| $N = 25$ | | | | | | | |
| 0.25;0.50 | 0.36 (0) | 0.38 (0) | 0.20 (8) | 0.40 (1) | 0.44 (0) | 0.35 (1) | 0.32 (1) |
| 0.25;1.50 | 0.39 (0) | 0.28 (3) | 0.26 (5) | 0.38 (2) | 0.36 (1) | 0.30 (4) | 0.32 (1) |
| 0.50;0.50 | 0.67 (2) | 0.70 (1) | 0.59 (9) | 0.93 (0) | 0.84 (0) | 0.75 (0) | 0.62 (6) |
| 0.50;1.50 | 0.65 (3) | 0.72 (1) | 0.70 (1) | 0.99 (0) | 0.98 (0) | 0.82 (2) | 0.60 (7) |
| 0.75;0.50 | 0.94 (1) | 0.94 (1) | 0.84 (7) | 0.99 (0) | 0.98 (0) | 0.91 (1) | 0.90 (3) |
| 0.75;1.50 | 0.94 (1) | 0.94 (2) | 0.84 (7) | 1.00 (0) | 0.99 (0) | 0.96 (0) | 0.93 (4) |
| $N = 50$ | | | | | | | |
| 0.25;0.50 | 0.31 (1) | 0.31 (1) | 0.25 (3) | 0.24 (5) | 0.31 (0) | 0.30 (0) | 0.26 (1) |
| 0.25;1.50 | 0.29 (0) | 0.13 (5) | 0.24 (0) | 0.20 (1) | 0.14 (1) | 0.10 (5) | 0.20 (0) |
| 0.50;0.50 | 0.60 (2) | 0.60 (2) | 0.52 (7) | 0.74 (0) | 0.74 (0) | 0.67 (0) | 0.57 (4) |
| 0.50;1.50 | 0.61 (0) | 0.71 (0) | 0.57 (4) | 0.97 (0) | 0.94 (0) | 0.88 (0) | 0.55 (6) |
| 0.75;0.50 | 0.87 (1) | 0.87 (1) | 0.87 (2) | 0.99 (0) | 0.95 (0) | 0.91 (0) | 0.83 (8) |
| 0.75;1.50 | 0.86 (5) | 0.88 (3) | 0.79 (5) | 1.00 (0) | 1.00 (0) | 0.96 (0) | 0.86 (4) |

TABLE 5 - *Total Flow Time*

4-MACHINE SYSTEM

| $Z; R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| | | | $N = 25$ | | | | |
| 0.25;0.50 | 4.28 (1) | 4.53 (0) | 4.41 (1) | 4.93 (0) | 4.68 (0) | 4.46 (0) | 4.05 (8) |
| 0.25;1.50 | 4.36 (5) | 4.76 (1) | 4.72 (1) | 4.88 (0) | 5.03 (0) | 4.78 (0) | 4.45 (4) |
| 0.50;0.50 | 4.36 (0) | 4.36 (0) | 4.06 (4) | 5.05 (0) | 4.56 (0) | 4.41 (1) | 4.05 (5) |
| 0.50;1.50 | 4.22 (0) | 4.33 (0) | 4.13 (4) | 5.23 (0) | 4.57 (0) | 4.33 (0) | 4.01 (6) |
| 0.75;0.50 | 4.12 (0) | 4.10 (0) | 3.65 (8) | 6.03 (0) | 4.26 (0) | 4.31 (0) | 3.93 (2) |
| 0.75;1.50 | 4.26 (1) | 4.24 (1) | 3.92 (6) | 5.24 (0) | 4.31 (0) | 4.41 (0) | 4.15 (3) |
| | | | $N = 50$ | | | | |
| 0.25;0.50 | 7.72 (1) | 8.26 (0) | 7.44 (4) | 8.37 (0) | 8.21 (0) | 8.21 (0) | 7.57 (5) |
| 0.25;1.50 | 7.83 (5) | 8.17 (0) | 7.78 (6) | 8.57 (0) | 8.27 (0) | 8.50 (0) | 8.00 (2) |
| 0.50;0.50 | 7.54 (0) | 7.56 (0) | 7.08 (5) | 8.42 (0) | 7.94 (0) | 7.94 (0) | 7.15 (5) |
| 0.50;1.50 | 7.54 (0) | 7.97 (0) | 7.30 (4) | 8.61 (0) | 8.14 (0) | 7.95 (0) | 7.44 (6) |
| 0.75;0.50 | 7.23 (1) | 7.18 (1) | 6.70 (7) | 9.61 (0) | 7.69 (0) | 7.77 (0) | 7.05 (2) |
| 0.75;1.50 | 7.23 (1) | 7.18 (1) | 6.70 (7) | 9.61 (0) | 7.69 (0) | 7.77 (0) | 7.05 (2) |

TABLE 6 - *Total Flow Time*

8-MACHINE SYSTEM

| $Z; R$ | *Scheduling Rule* | | | | | | |
|---|---|---|---|---|---|---|---|
| | *RM-1* | *RM-Iter* | *FSD* | *SB1* | *SB2* | *Botflow* | *MFS* |
| | | | $N = 25$ | | | | |
| 0.25;0.50 | 2.89 (0) | 2.93 (0) | 2.63 (6) | 3.26 (0) | 3.22 (0) | 2.98 (0) | 2.77 (4) |
| 0.25;1.50 | 3.07 (1) | 3.14 (0) | 2.68 (4) | 3.35 (1) | 3.19 (0) | 3.15 (0) | 2.94 (4) |
| 0.50;0.50 | 2.96 (0) | 2.96 (0) | 2.56 (9) | 3.71 (0) | 3.21 (0) | 3.03 (0) | 2.81 (1) |
| 0.50;1.50 | 2.94 (1) | 2.99 (1) | 2.74 (6) | 4.11 (0) | 3.12 (0) | 3.01 (0) | 2.88 (3) |
| 0.75;0.50 | 2.92 (1) | 2.92 (1) | 2.25 (8) | 4.36 (0) | 3.00 (0) | 2.93 (0) | 2.81 (1) |
| 0.75;1.50 | 2.91 (0) | 2.89 (0) | 2.31 (9) | 3.90 (0) | 2.92 (0) | 2.92 (0) | 2.80 (1) |
| | | | $N = 50$ | | | | |
| 0.25;0.50 | 4.87 (1) | 4.93 (0) | 4.65 (4) | 4.97 (0) | 5.11 (0) | 4.92 (0) | 4.64 (6) |
| 0.25;1.50 | 4.84 (3) | 5.15 (0) | 4.77 (3) | 5.29 (0) | 4.65 (4) | 5.14 (0) | 4.81 (3) |
| 0.50;0.50 | 4.65 (2) | 4.65 (2) | 4.43 (5) | 5.36 (0) | 5.01 (0) | 4.90 (0) | 4.55 (3) |
| 0.50;1.50 | 4.79 (0) | 4.84 (0) | 4.49 (5) | 5.55 (1) | 4.98 (0) | 4.90 (0) | 4.67 (4) |
| 0.75;0.50 | 4.63 (2) | 4.60 (2) | 4.45 (4) | 7.36 (0) | 4.84 (0) | 4.84 (0) | 4.52 (4) |
| 0.75;1.50 | 4.59 (0) | 4.58 (1) | 3.77 (9) | 7.71 (0) | 4.75 (0) | 4.89 (0) | 4.58 (0) |

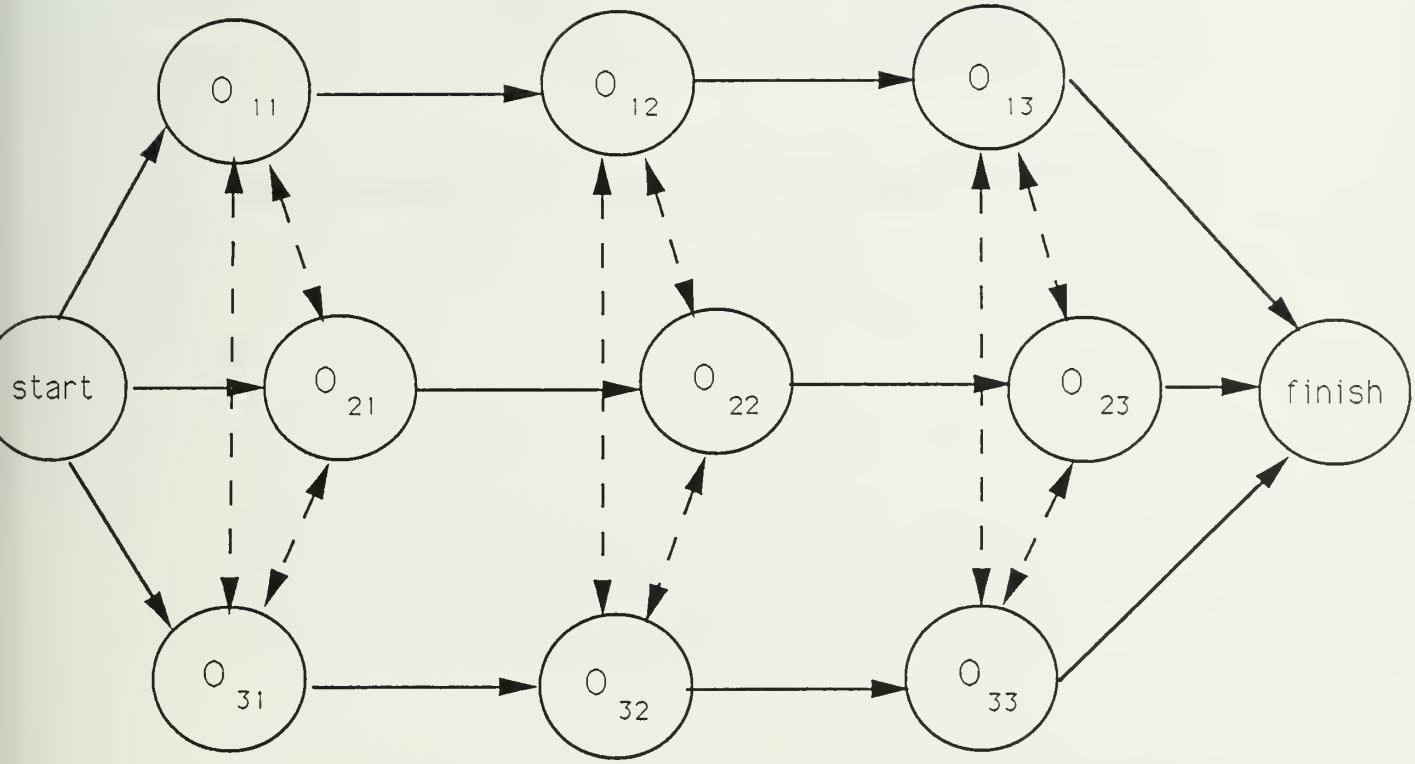


Figure 1: Network Representation of the Flow Shop Sequencing Problem

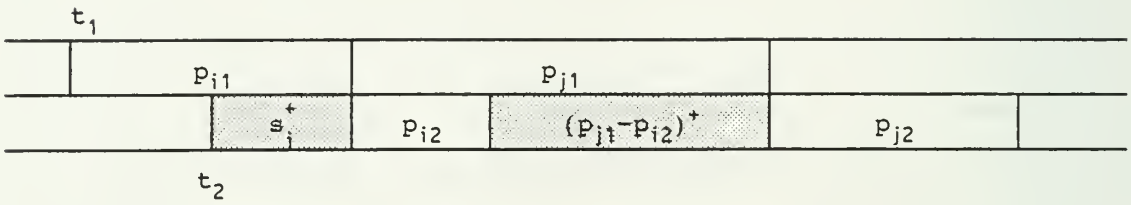


Figure 2a: 2–Machine Flow Shop: Schedule with Leading Idle Time

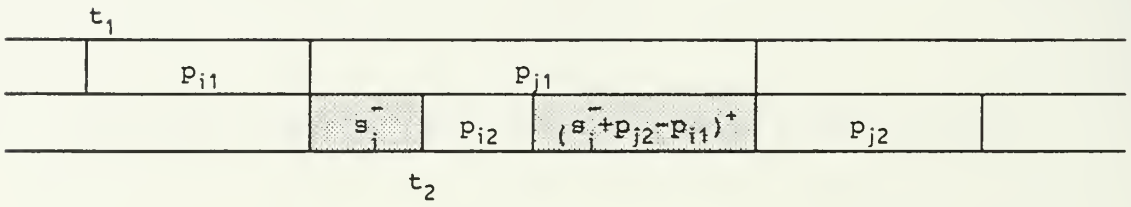


Figure 2b: 2–Machine Flow Shop: Schedule with Trailing Idle Time

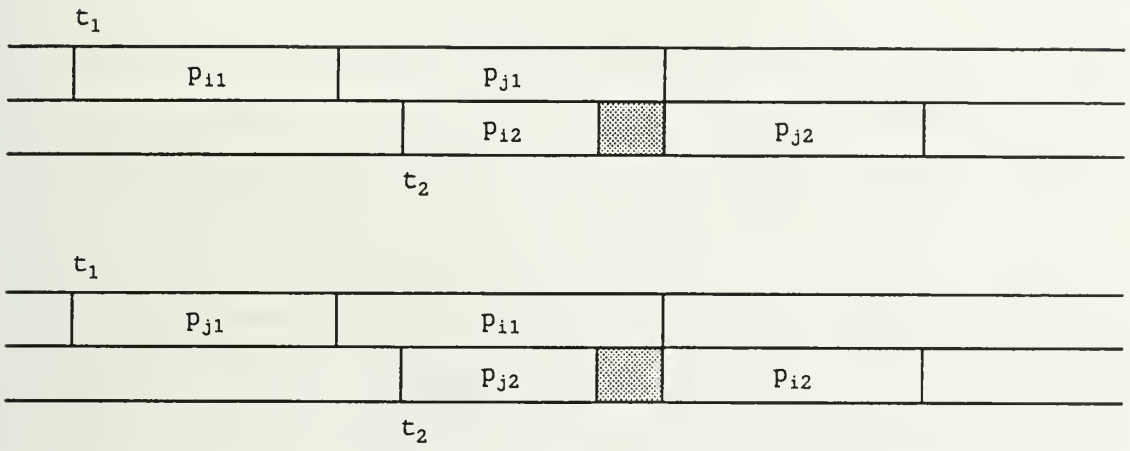


Figure 3: 2–Machine Flow Shop: Configuration 1

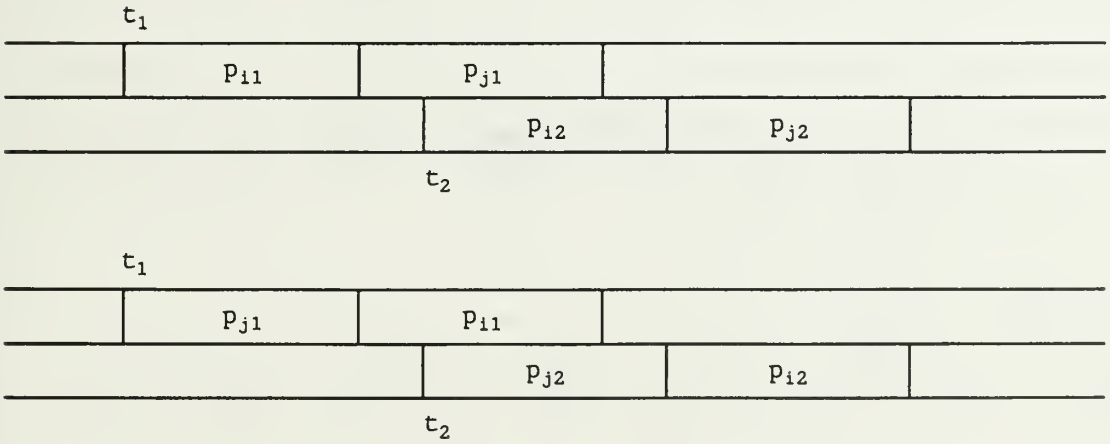


Figure 4: 2–Machine Flow Shop: Configuration 2

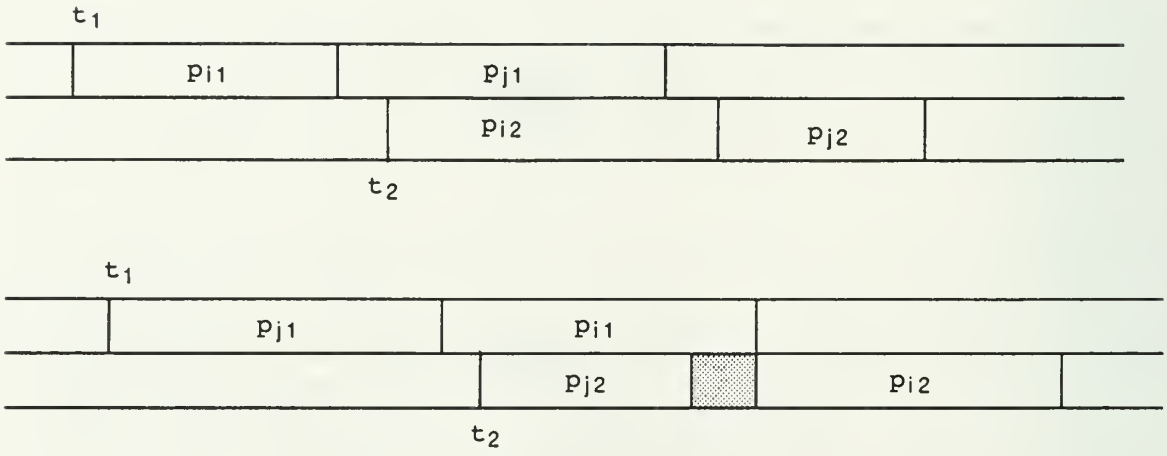


Figure 5: 2–Machine Flow Shop: Configuration 3

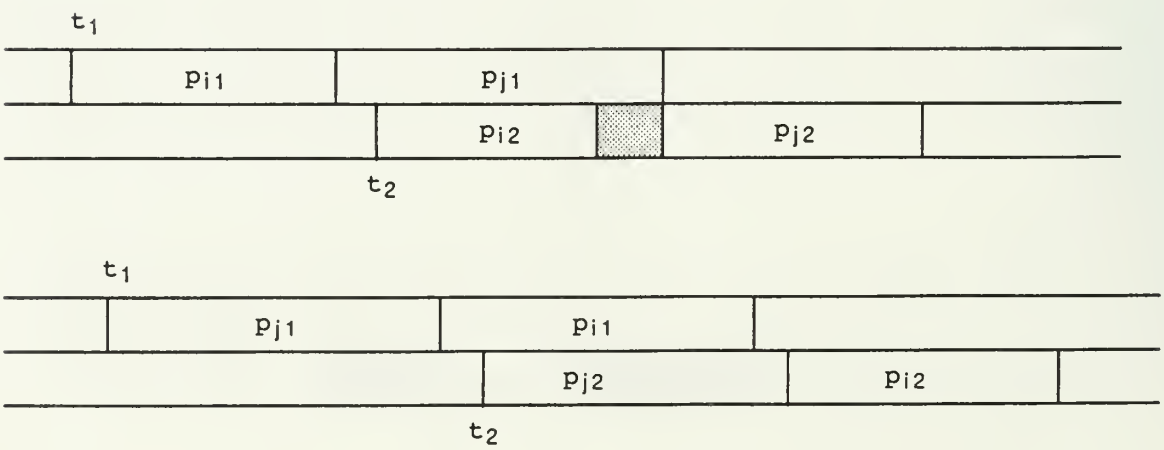


Figure 6: 2–Machine Flow Shop: Configuration 4

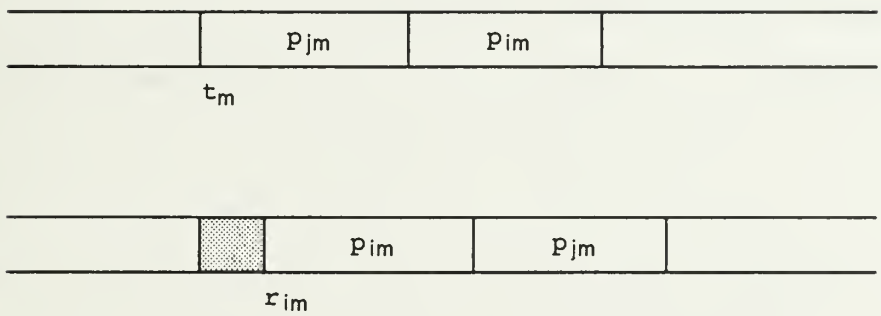


Figure 7: Impact of Forced Idle Time

## APPENDIX 1

### Proof of Remark 1

Note that in this case, there is no trailing idle time with either of the jobs, i.e., $S_i^- + p_{j1} - p_{i2} = S_j^- + p_{i1} - p_{j2} = 0$. The proof is based on considering two levels of problem instances that are mutually exclusive and collectively exhaustive. At the first level, we consider four subcases:

A. No leading idle times with either of the jobs, i.e., $S_i^+ = S_j^+ = 0$.

B. Leading idle time following both jobs, i.e., $S_i^+ > 0$, $S_j^+ > 0$.

C. Leading idle time following i but not j, i.e., $S_i^+ > 0$, $S_j^+ = 0$.

D. Leading idle time following j but not i, i.e., $S_i^+ = 0$, $S_j^+ > 0$.

According to Remark 1, i precedes j in an optimal schedule if and only if

$$\max\{\max(t_1 + p_{i1}, t_2) + p_{i2}, d_i\} + \max(t_1 + p_{j1}, t_2)$$
$$\leq \max\{\max(t_1 + p_{j1}, t_2) + p+j2, d_j\} + \max(t_1 + p_{j1}, t_2).$$

Within each level, we consider all feasible scenarios. Let $\sigma_{ij}$ denote the subsequence in which $i \prec j$, and $\sigma_{ji}$ be the alternative subsequence in which $j \prec i$. Let $T_i$ and $T_j$ ($T_i'$ and $T_j'$) denote the tardiness of i and j in $\sigma_{ij}$ ($\sigma_{ji}$). Let $T_{ij} = T_i + T_j$ and $T_{ji} = T_j' + T_i'$. Then, it suffices to show that $\Delta T = T_{ij} - T_{ji} \leq 0$ if and only if (AO) is satisfied.

We use a 4-tuple $(v_1v_2v_3v_4)$ to represent all possible scenarios that need to be considered, where

$v_1 = 1$ if $T_i > 0$, and 0 otherwise,

$v_2 = 1$ if $T_j > 0$, and 0 otherwise,

$v_3 = 1$ if $T_j' > 0$, and 0 otherwise,

$v_4 = 1$, if $T_i' > 0$, and 0 otherwise.

There are sixteen scenarios possible depending upon whether or not i and j are tardy in the two sequences. These can be represented as (0000), (0001), …, (1111). However, note that $T_i \leq T_i'$ and $T_j' \leq T_j$, which implies that i) if $v_1 = 1$ then $v_4 = 1$, and ii) if $v_3 = 1$, then $v_2 = 1$. Consequently, the only scenarios that we need to consider are (0000), (0001), (0101), (0110), (0111), (0100), (1001), (1101), and (1111).

Among these, (0000) is handled trivially. It is easy to see that $\Delta T > 0$ in (0100) and (0110). It is sufficient to show that these scenarios are infeasible if (A0) is true. We prove this to be so for (0100); the proof for (0110) is similar, and therefore, it is omitted. Similarluy, it can be seen that $\Delta T < 0$ in (0001) and (1001). It is sufficient to show that (A0) is automatically true in these scenarios. The proof of this result is similar to that of the infeasibility of (0100) referredt o above, and it is omitted.

The remaining four scenarios are less obvious; these are delat with individually. In summary, we deal with the following five scenarios:

1. (1111): Both i and j are late in either position,
2. (0101): Both i and j are early if scheduled first, late otherwise,
3. (0111): i is early if scheduled first, late otherwise; j is late in either position,

4. (1101): j is early if scheduled first, late otherwise; i is late in either position, and
5. (0100): j is early if scheduled first, late otherwise; i is early in either position.

I. $\underline{s_i^+ = s_j^+ = 0}$

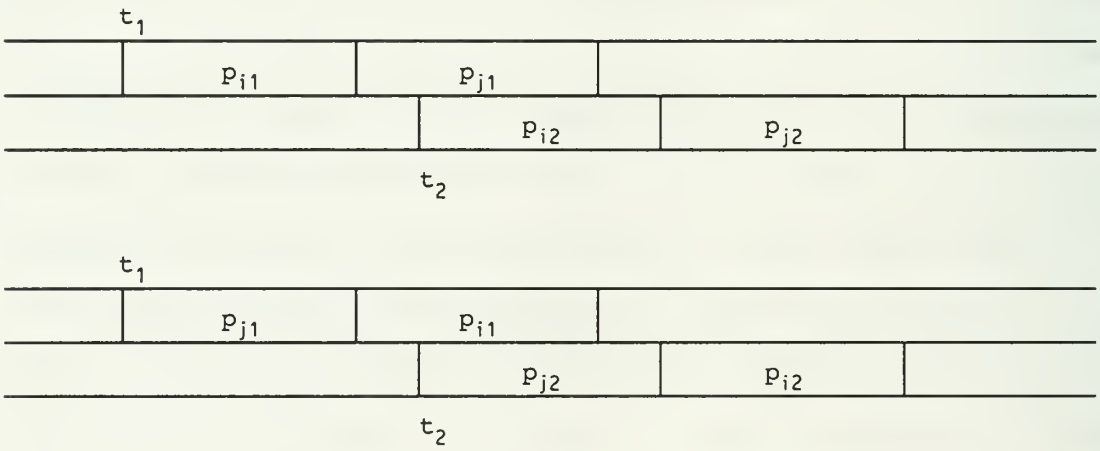


Figure 8: Subcase I

Note that in this subcase,

$$\begin{aligned}\Delta T &= T_{ij} - T_{ji} \\ &= [(t_2+p_{i2}-d_i)^+ + (t_2+p_{i2}+p_{j2}-d_j)^+] \\ &\quad - [(t_2+p_{j2}-d_j)^+ + (t_2+p_{i2}+p_{j2}-d_i)^+]\end{aligned}$$

and according to Remark 1, i precedes j if and only if

$$\max\{t_2+p_{i2}, d_i\} \le \max\{t_2+p_{j2}, d_j\}. \qquad (A0)$$

We need consider only the following instances:

i) Both i and j are late in both positions:

In this case,

$$T_{ij} = (t_2 + p_{i2} - d_i) + (t_2 + p_{i2} + p_{j2} - d_j),$$
$$T_{ji} = (t_2 + p_{j2} - d_j) + (t_2 + p_{i2} + p_{j2} - d_i)$$

and

$$\Delta T = T_{ij} - T_{ji} = p_{i2} - p_{j2}. \tag{1}$$

As $t_2 \geq t_1 + \max\{p_{i1}, p_{j1}\}$, $t_2 + p_{i2} \geq d_i$, and $t_2 + p_{j2} \geq d_j$, (AO) reduces to the condition that i precedes j if

$$p_{i2} \leq p_{j2}. \tag{A1}$$

From (1) and (A1), it follows that i precedes j if and only if $\Delta T < 0$.

ii) Both i and j are early if scheduled first, late otherwise:

In this case,

$$\Delta T = (t_2 + p_{i2} + p_{j2} - d_j) - (t_2 + p_{j2} + p_{i2} - d_i) = d_i - d_j \tag{2}$$

and (AO) reduces to the condition that i precedes j if

$$d_i \leq d_j. \tag{A2}$$

From (2) and (A2) it follows that i precedes j if and only if $\Delta T < 0$.

iii) <u>i early if scheduled first, late otherwise; j late in either position</u>:

In this case,

$$\begin{aligned}\Delta T &= (t_2+p_{i2}+p_{j2}-d_j) - (t_2+p_{j2}-d_j) - (t_2+p_{i2}+p_{j2}-d_i) \\ &= d_i - (t_2+p_{j2}).\end{aligned} \tag{3}$$

From (A0), i precedes j if

$$d_i \le t_2 + p_{j2} \tag{A3}$$

and Remark 1 is both necessary and sufficient for local optimality.

iv) <u>j early if scheduled first; late otherwise; i late in either position</u>:

In this case

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_2+p_{i2}+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (t_2+p_{i2}) - d_j.\end{aligned} \tag{4}$$

From (A0), i precedes j if

$$t_2 + p_{i2} \le d_j, \tag{A4}$$

and Remark 1 is both necessary and sufficient for local optimality.

v) <u>j is early if scheduled first, late otherwise; i is early in either position</u>:

We need to show that this case is infeasible if DR(2) favors i over j.

In this case,

$$\Delta T = t_2 + p_{i2} + p_{j2} - d_j > 0. \tag{5}$$

But because i is early in the latter position

$$t_2 + p_{i2} + p_{j2} - d_i < 0. \tag{6}$$

(5) and (6) imply that $d_i > d_j$. (7)

But since $P_{i2} \leq P_{j2}$ favors i, from (A0) we have $d_i \leq d_j$. This contradicts (7) and the desired infeasibility is established.

II. <u>$s_i^+ > 0$; $s_j^+ = 0$</u>

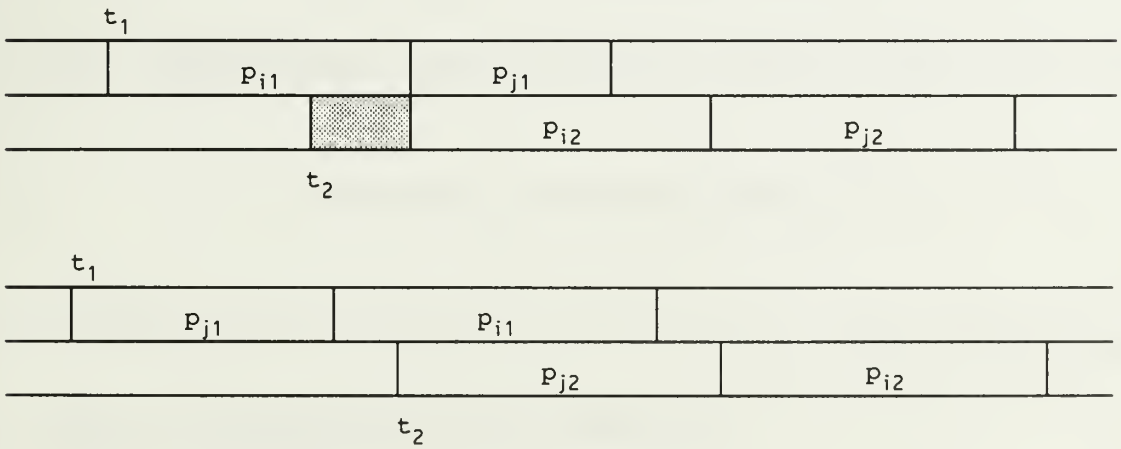


Figure 9: Subcase II

Note that in this subcase,

$$\Delta T = [(t_1+p_{i1}+p_{i2}-d_i)^+ + (t_1+p_{i1}+p_{i2}+p_{j2}-d_j)^+]$$
$$- [(t_2+p_{j2}-d_j)^+ + (t_2+p_{j2}+p_{i2}-d_i)^+]$$

and according to Remark 1, i precedes j if

$$\max\{t_1+p_{i1}+p_{i2}, d_i\} + (t_1+p_{i1}) \le \max(t_2+p_{j2}, d_j) + t_2. \qquad (B0)$$

i) Both i and j are late in both positions:

$$\begin{aligned}\Delta T = T_{ij} - T_{ji} &= (t_1+p_{i1}+p_{i2}-d_i) + (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (t_2+p_{i1}+p_{i2}) + (t_1+p_{i1}) - (t_2+p_{j2}) - t_2.\end{aligned} \qquad (8)$$

From (B0), i precedes j if and only if

$$(t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) \le (t_2+p_{j2}) + t_2.$$

Hence, Remark 1 is both necessary and sufficient for local optimality.

ii) Both i and j are early if scheduled first, late otherwise:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \\ &= d_i + (t_1+p_{i1}) - d_j - t_2.\end{aligned} \qquad (9)$$

From (B0), i precedes j if

$$d_i + (t_1+p_{i1}) \le d_j + t_2.$$

Hence, Remark 1 is both necessary and sufficient for local optimality.

iii) <u>i early if scheduled first, late otherwise;
j late in either position</u>:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) - (t_2+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}+p_{i2}-d_i) \\ &= d_i + (t_2+p_{i1}) - (t_2+p_{j2}) - t_2.\end{aligned} \tag{10}$$

From (B0), i precedes j if

$$d_i + (t_1+p_{i1}) \leq (t_2+p_{j2}) - t_2.$$

Hence, Remark 1 is both necessary and sufficient for local optimality.

iv) <u>j early if scheduled first, late otherwise;
i late in either position</u>:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}-d_i) + (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) - d_j - t_2.\end{aligned} \tag{11}$$

From (B0), i precedes j if

$$(t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) \leq d_j + t_2.$$

Hence, Remark 2 is both necessary and sufficient for local optimality.

v) <u>j early if scheduled first, later otherwise;
i early in either position</u>:

We need to show that this case is infeasible. We have

$$\Delta T = t_1 + p_{i1} + p_{i2} + p_{j2} - d_j > 0. \tag{12}$$

But since i is early in the latter position

$$t_2 + p_{j2} + p_{i2} - d_i < 0. \tag{13}$$

From (12) and (13), we have

$$d_i + (t_1 + p_{i1}) - d_j - t_2 > 0. \tag{14}$$

But since i precedes j, from (B0) we have

$$d_i + (t_1 + p_{i1}) \leq d_j + t_2.$$

This contradicts (14) and the desired infeasibility is established.

III. $\underline{s_i^+ = 0;\ s_j^+ > 0}$

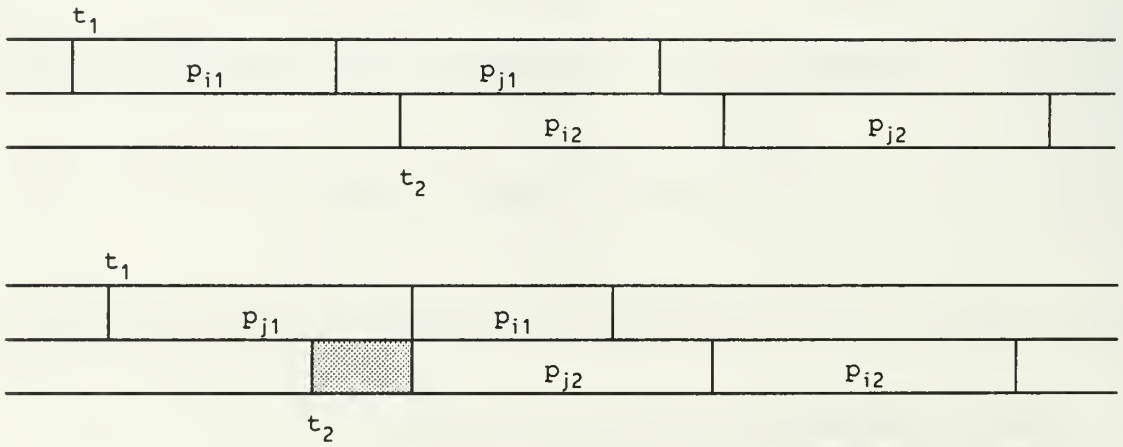


Figure 10: Subcase III

In this subcase,

$$\Delta T = [(t_2+p_{i2}-d_i)^+ + (t_2+p_{i2}+p_{j2}-d_j)^+]$$
$$- [(t_1+p_{j1}+p_{j2}-d_j)^+ + (t_1+p_{j1}+p_{j2}+p_{i2}-d_i)^+]$$

and according to Remark 1, i precedes j if and only if

$$\max\{t_2+p_{i2}, d_i\} + t_2 \le \max\{t_1+p_{j1}+p_{j2}, d_j\} + (t_1+p_{j1}). \qquad (C0)$$

i) Both i and j are late in both positions:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_2+p_{i2}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= (t_2+p_{i2}) + t_2 - (t_1+p_{j1}+p_{j2}) - (t_1+p_{j1}).\end{aligned} \qquad (15)$$

From (C0), i precedes j if

$$(t_2+p_{i2}) + t_2 \le (t_1+p_{j1}+p_{j2}) + (t_1+p_{j1}). \qquad (C9)$$

Hence, Remark 1 is both necessary and sufficient for local optimality.

ii) Both i and j are early if scheduled first, late otherwise:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}+p_{j2}-d_j) - (t_1+p_{j1}+p_{i2}-d_i) \\ &= d_i + t_2 - (t_1+p_{j1}) - d_j.\end{aligned} \qquad (16)$$

From (C0) and (16), i precedes j if and only if $\Delta T \le 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

iii) i is early if scheduled first, late otherwise;
j is late in both positions:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= d_i + t_2 - (t_1+p_{j1}+p_{j2}) - (t_1+p_{j1}).\end{aligned} \tag{17}$$

From (C0) and (17), i precedes j if and only if $\Delta T \leq 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

iv) j is early if scheduled first, late otherwise;
i is late in both positions:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_2+p_{i2}+p_{j2}-d_j) \\ &\quad - (t+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= (t_2+p_{i2}) + t_2 - d_j - (t_1+p_{j1}).\end{aligned} \tag{18}$$

From (C0) and (18), i precedes j if and only if $\Delta T \leq 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

v) j is early if scheduled first; late otherwise;
i is early in both positions:

We need to show that this instance is infeasible.

In this case

$$\Delta T = t_2 + p_{i2} + p_{j2} - d_j > 0. \tag{19}$$

But since i is early in the latter position

$$t_1 + p_{j1} + p_{j2} + p_{i2} - d_i < 0. \tag{20}$$

From (19) and (20)

$$d_i + t_2 - d_j - (t_1 + p_{j1}) > 0. \tag{21}$$

But, since i precedes j, it follows from (C0) that

$$d_i + t_2 \leq d_j + (t_1 + p_{j1})$$

which contradicts (21) and the required infeasibility is established.

IV. $\underline{s_i^+ > 0,\ s_j^+ > 0}$

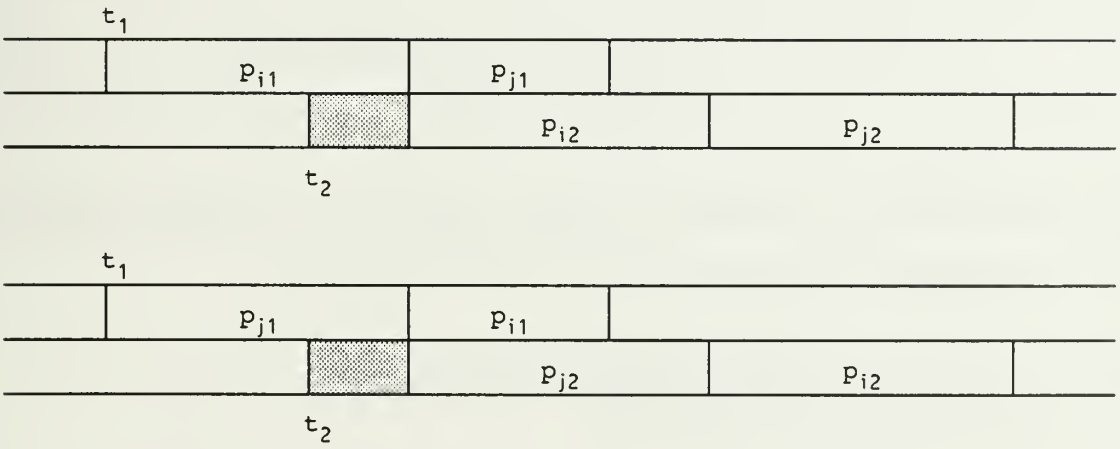


Figure 11: Subcase IV

In this subcase,

$$\Delta T = [(t_1 + p_{i1} + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{i2} + p_{j2} - d_j)^+]$$
$$- [(t_1 + p_{j1} + p_{j2} - d_j)^+ + (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i)^+]$$

and according to Remark 1, i precedes j if and only if

$$\max\{t_1+p_{i1}+p_{i2}, d_i\} + (t_1+p_{i1}) \le \max\{t_1+p_{j1}+p_{j2}, d_j\} + (t_1+p_{j1}). \qquad (D0)$$

i) Both i and j are late in both positions:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2} - d_i) + (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}-d_j) + (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= (t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) - (t_1+p_{j1}+p_{j2}) - (t_1+p_{j1}).\end{aligned} \qquad (22)$$

From (D0) and (22), i precedes j if and only if $\Delta T \le 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

ii) Both i and j are early if scheduled first, late otherwise:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= d_i + (t_1+p_{i1}) - d_j - (t_1+p_{j1}).\end{aligned} \qquad (23)$$

From (D0) and (23), i precedes j if and only if $\Delta T \le 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

iii) i is early if scheduled first, late otherwise; j is late in both positions:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}+p_{j2}-d_j) + (t_1+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= d_i + (t_1+p_{i1}) - (t_1+p_{j1}+p_{j2}) - (t_1+p_{j1}).\end{aligned} \qquad (24)$$

From (D0) and (24), i precedes j if and only if $\Delta T \le 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

iv) j is early if scheduled first, late otherwise;
i is late in both positions:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}+p_{j2}-d_j)+(t_1+p_{i1}+p_{i2}-d_i)\\ &\quad -(t_1+p_{j1}+p_{j2}+p_{i2}-d_i)\\ &= (t_1+p_{i1}+p_{i2})+(t_1+p_{i1})-d_j-(t_1+p_{j1}).\end{aligned} \tag{25}$$

From (D0) and (25), i precedes j if and only if $\Delta T \leq 0$. Hence, Remark 1 is both necessary and sufficient for local optimality.

v) j is early if scheduled first, late otherwise;
i is early in both positions:

As before, we show that if i precedes j according to Remark 1, then this instance is not feasible.

Note that, in this case

$$\Delta T = t_1 + p_{i1} + p_{i2} + p_{j2} - d_j > 0 \tag{26}$$

But since i is early in the latter position

$$t_1 + p_{j1} + p_{j2} + p_{i2} - d_i < 0. \tag{27}$$

From (26) and (27), it follows that

$$d_i + (t_1+p_{i1}) - d_j - (t_1+p_{j1}) > 0. \tag{28}$$

However, since i precedes j according to Remark 1

$$d_i + (t_1 + p_{i1}) \leq d_j + (t_1 + p_{j1}).$$

But this contradicts (28) and the infeasibility of this instance is established.

## APPENDIX 2

### Proof of Remark 2

Note that in this case, there are trailing idle times following both jobs, i.e., $S_i^- + p_{j1} - p_{i2} > 0$; $S_j^- + p_{i1} - p_{j2} > 0$. As with the proof of Remark 1, this proof is based on considering four subcases and five scenarios within each subcase.

According to Remark 2, i precedes j if and only if

$$\max\{t_1 + p_{i1}, d_i - p_{i2}\} \leq \max\{t_1 + p_{j1}, d_j - p_{j2}\}. \qquad (E0)$$

Consider the following subcases:

I. $\underline{s_i^+ = s_j^+ = 0}$

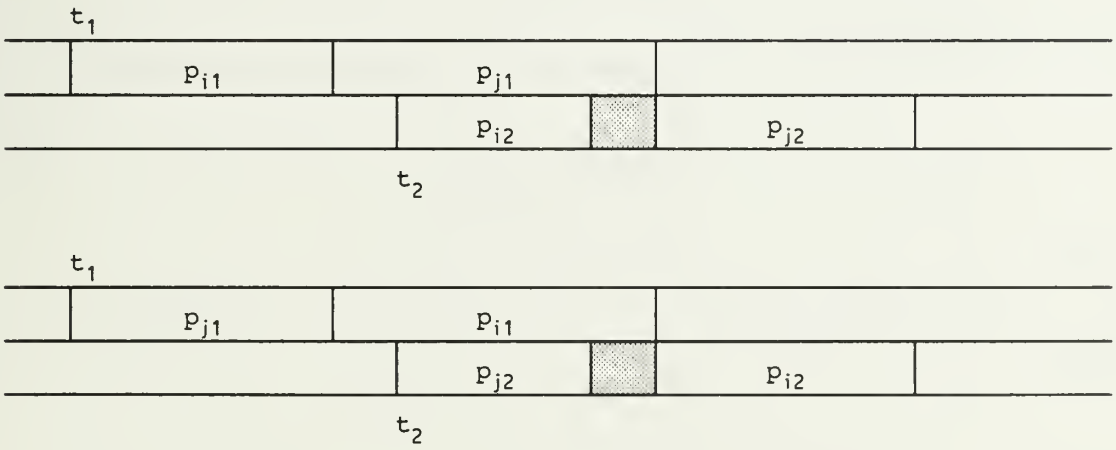


Figure 12: Subcase I

$$\begin{aligned}\Delta T &= T_{ij} - T_{ji} \\ &= [(t_2 + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j)^+] \\ &\quad - [(t_2 + p_{j2} - d_j)^+ + (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i)^+].\end{aligned}$$

Consider the following instances:

i) Both i and j are late in both positions:

$$\begin{aligned}\Delta T &= (t_2 + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_2 + p_{j2} - d_j) - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= 0.\end{aligned}$$

Both sequences are equally good, and without losing any generality Remark 2 is necessary and sufficient for local optimality.

ii) Both i and j are early if scheduled first, late otherwise:

$$\begin{aligned}\Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= (d_i - p_{i2}) - (d_j - p_{j2})\end{aligned} \tag{29}$$

and (E0) reduces to

$$d_i - p_{i2} \leq d_j - p_{j2}. \tag{E1}$$

From (E1) and (29), i precedes j if and only if $\Delta T < 0$ and Remark 2 holds.

iii) i is early if scheduled first, late otherwise;
j is late in either position:

Note that, in this case, (E0) reduces to

$$d_i - p_{i2} \leq t_1 + p_{j1},$$
$$\text{or} \quad d_i \leq t_1 + p_{j1} + p_{i2} \leq t_2 + p_{i2}.$$

But this contradicts the assumption that i is early if scheduled first.

It follows that in this instance Remark 2 is infeasible.

iv) j is early if scheduled first; late otherwise;
i is late in either position:

$$\begin{aligned}\Delta T &= (t_2 + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= t_2 - (d_j - p_{j2}) < 0.\end{aligned}$$

Also note that in this instance (E0) reduces to

$$t_1 + p_{i1} \leq d_j - p_{j2}$$
$$\text{or} \quad d_j \geq t_1 + p_{i1} + p_{j2}. \qquad \text{(E2)}$$

But since j is early in the first position

$$d_j \geq t_2 + p_{j2} \geq t_1 + p_{i1} + p_{j2}$$

and (E2) is always satisfied.

Hence, in this instance $P_{i1} \leq P_{j1}$ always, and $\Delta T$ is always negative and Remark 2 holds.

v) <u>j is early if scheduled first, late otherwise;
i is early in either position</u>:

$$\Delta T = t_1 + p_{i1} + p_{j1} + p_{j2} - d_j > 0. \tag{30}$$

But because, i is early in the later position

$$t_1 + p_{j1} + p_{i1} + p_{i2} - d_i < 0. \tag{31}$$

(30) and (31) imply that

$$p_{j2} - d_j - p_{i2} + d_i > 0. \tag{32}$$

However, since $P_{i1} \leq P_{j1}$

$$d_i - p_{i2} \leq d_j - p_{j2}.$$

But this contradicts (32) and the infeasibility of this instance is established.

II. $s_i^+ > 0,\ s_j^+ = 0$

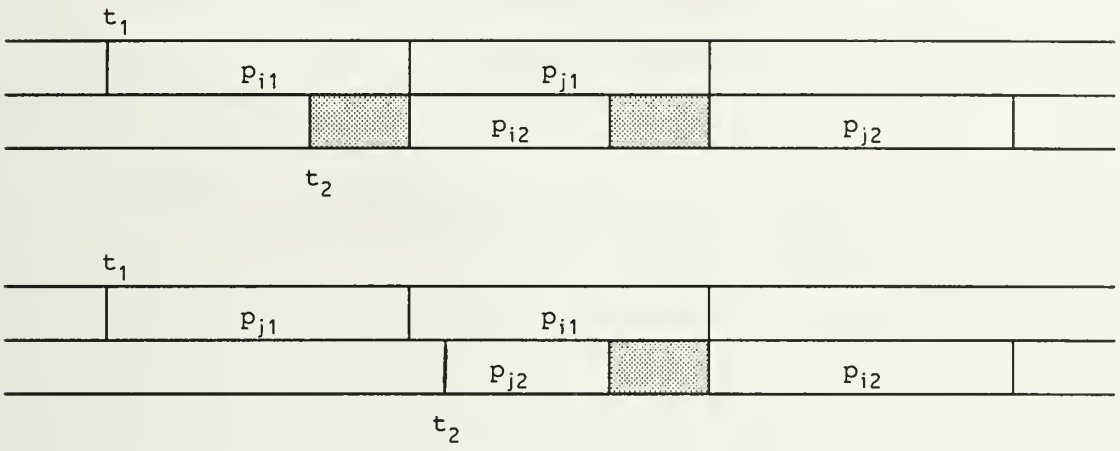


Figure 13: Subcase II

$$\Delta T = [(t_1 + p_{i1} + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j)^+]$$
$$- [(t_2 + p_{j2} - d_j)^+ + (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i)^+] .$$

Consider the following instances:

i) Both i and j are late in either position:

Note that, in this case (E0) reduces to

$$t_1 + p_{i1} \leq t_1 + p_{j1} .$$

But this is infeasible because $t_1 + p_{i1} > t_2 \geq t_2 + p_{j1}$ as seen in the figure. Hence, in this instance $P_{i1} \not\leq P_{j1}$.

ii) Both i and j are early if scheduled first, late otherwise:

$$\begin{aligned} \Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= (d_i - p_{i2}) - (d_j - p_{j2}) . \end{aligned} \tag{33}$$

And (E0) reduces to

$$d_i - p_{i2} \leq d_j - p_{j2}. \qquad \text{(E3)}$$

From (33) and (E3), i precedes j if and only if $\Delta T < 0$ and Remark 2 holds.

iii) <u>i early if scheduled first, late otherwise;<br>j late in either position</u>:

In this case (E0) reduces to

$$d_i - p_{i2} \leq t_1 + p_{j1}$$

$$\text{or} \quad d_i \leq t_1 + p_{j1} + p_{i2}. \qquad \text{(E4)}$$

Because i is early in the first position

$$d_i \geq t_2 + p_{i2} > t_1 + p_{j1} + p_{i2}$$

which contradicts (E4). Hence, in this instance, $P_{i1} \not\leq P_{j1}$.

iv) <u>j early if scheduled first, late otherwise;<br>i late in either position</u>

$$\begin{aligned} \Delta T &= (t_1 + p_{i1} + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= (t_1 + p_{i1}) - (d_j - p_{j2}). \end{aligned} \qquad \text{(34)}$$

And (E0) reduces to

$$t_1 + p_{i1} \leq d_j - p_{j2}. \qquad \text{(E5)}$$

From (34) and (E4), i precedes j if and only if $\Delta T < 0$.

v) <u>j early if scheduled first; i early in either position</u>:

In this case, (E0) reduces to

$$d_i - p_{i2} \leq d_j - p_{j2}. \qquad \text{(E6)}$$

But $d_i - p_{i2} > t_1 + p_{j1} + p_{i1}$ because i is early in the later position

$> d_j - p_{j2}$ because j is late in the later position.

This contradicts (E6). Hence, in this instance, $P_{i1} \not\leq P_{j1}$.

III. <u>$s_i^+ = 0$, $s_j^+ > 0$</u>

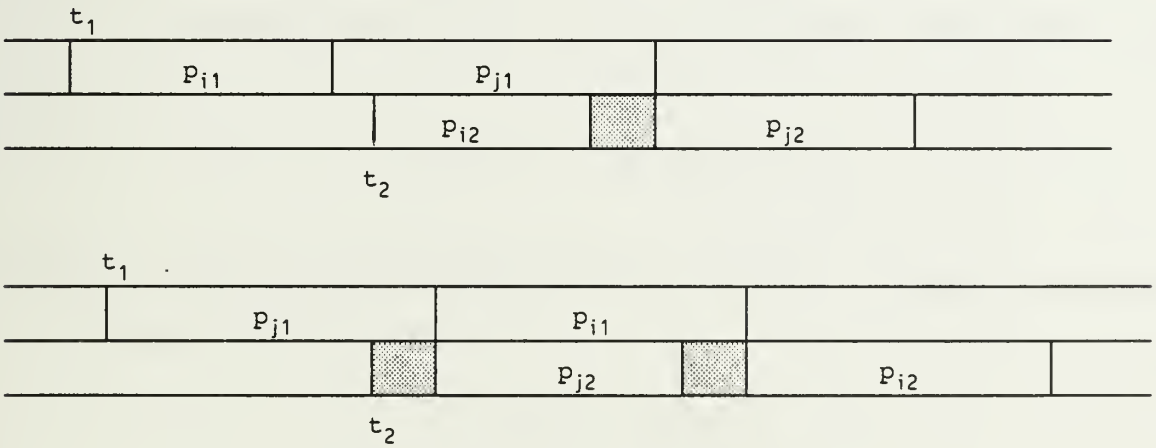


Figure 14: Subcase III

$$\Delta T = [(t_2 + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j)^+]$$
$$- [(t_1 + p_{j1} + p_{j2} - d_j)^+ + (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i)^+].$$

Consider the following instances:

i) Both i and j are late in either position:

$$\begin{aligned}\Delta T &= (t_2 + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{i1} + p_{i2} - d_i) \\ &= t_2 - (t_1 + p_{j1}) < 0.\end{aligned}$$

Note that in this case, (E0) reduces to

$$t_1 + p_{i1} \leq t_1 + p_{j1}$$

which is always true in this instance because

$$t_1 + p_{i1} \leq t_2 \leq t_1 + p_{j1}.$$

ii) Both i and j are early if scheduled first, late otherwise:

$$\Delta T = (d_i - p_{i2}) - (d_j - p_{j2}).$$

And (E0) reduces to

$$(d_i - p_{i2}) \leq (d_j - p_{j2}).$$

Hence, i precedes j according to Remark 2 if and only if $\Delta T < 0$.

iii) i early if scheduled first, late otherwise;
j late in either position:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{i1}+p_{i2}-d_i) \\ &= (d_i-p_{i2}) - (t_1+p_{j1}).\end{aligned}$$

(E0) reduces to

$$(d_i-p_{i2}) \leq (t_1+p_{j1}).$$

Hence, i precedes j according to Remark 2 if and only if $\Delta T < 0$.

iv) j early if scheduled first, late otherwise;
i late in either position:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{i1}+p_{i2}-d_i) \\ &= t_2 + p_{j2} - d_j < t_1 + p_{j1} + p_{j2} - d_j < 0\end{aligned}$$

because j is early in the first position.

In this case, (E0) reduces to

$$\max\{t_1 + p_{i1},\ d_i - p_{i2}\} \leq d_j - p_{j2}. \qquad \text{(E7)}$$

Note that $t_1 + p_{i1} < t_2$, and because i is late in the first position $d_i - p_{i2} < t_2$ as well.

Hence the LHS of (E7) is less than $t_2$. Clearly, $d_j - p_{j2} \geq t_1 + p_{j1} > t_2$, hence (E7) will always be satisfied, and Remark 2 holds.

v) <u>j early if scheduled first, late otherwise;
i early in either position</u>:

From (E0), i is favored in this case if

$$d_i - p_{i2} \leq d_j - p_{j2}. \tag{E8}$$

But, since i is early in the later position and j is not,

$$d_i - p_{i2} \geq t_1 + p_{i1} + p_{j1} > d_j - p_{j2}$$

which contradicts (E8). Hence, in this instance $P_{i1} \not\leq P_{ji}$.

IV. <u>$s_i^+ > 0$, $s_j^+ > 0$</u>

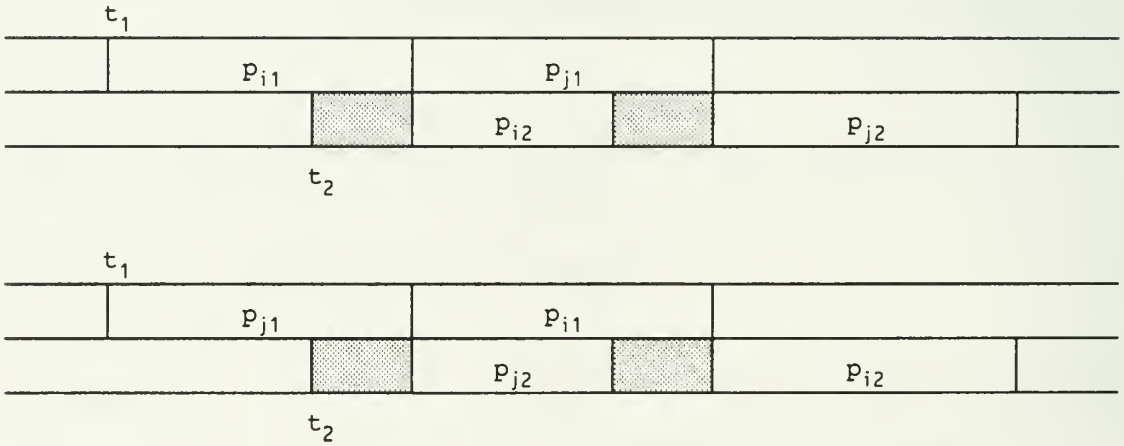


Figure 15: Subcase IV

$$\Delta T = [(t_1+p_{i1}+p_{i2}-d_i)^+ + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j)^+]$$
$$- [(t_1+p_{j1}+p_{j2}-d_j)^+ + (t_1+p_{j1}+p_{i1}+p_{i2}-d_i)^+].$$

Consider the following instances:

i) Both i and j late in both positions:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}-d_j) - (t_1+p_{j1}+p_{i1}+p_{i2}-d_i) \\ &= p_{i1} - p_{j1}.\end{aligned} \tag{35}$$

In this case, (E0) reduces to

$$t_1 + p_{i1} \leq t_1 + p_{j1}. \tag{E9}$$

From (35) and (E9), it can be seen that i precedes j if and only if $\Delta T < 0$.

ii) Both i and j early if scheduled first, late otherwise:

$$\Delta T = (d_i - p_{i2}) - (d_j - p_{j2}). \tag{36}$$

From (E0), in this case, i precedes j if

$$d_i - p_{i2} \leq d_j - p_{j2}. \tag{E10}$$

From (36) and (E10), i precedes j if and only if $\Delta T \leq 0$.

iii) i early if scheduled first, late otherwise;
j late in either position:

$$\Delta T = d_i - (t_1+p_{j1}+p_{i2}). \tag{37}$$

(E0) reduces to

$$d_i - p_{i2} \leq t_1 + p_{j1}. \tag{E11}$$

From (37) and (E11), i precedes j if and only if $\Delta T \leq 0$.

iv) <u>j early if scheduled first, late otherwise;
i late in either position</u>:

$$\Delta T = (t_1 + p_{i1} + p_{j2}) - d_j. \tag{38}$$

(E0) reduces to

$$t_1 + p_{i1} \leq d_j - p_{j2}. \tag{E12}$$

From (38) and (E12), i precedes j if and only if $\Delta T \leq 0$.

v) <u>j early if scheduled first, late otherwise;
i early in either position</u>:

(E0) reduces to

$$d_i - p_{i2} \leq d_j - p_{j2}. \tag{E13}$$

Since j is late in the later position

$$t_1 + p_{i1} + p_{j1} + p_{j2} - d_j > 0$$

$$\text{or} \quad d_j - p_{j2} < t_1 + p_{i1} + p_{j1} \leq d_i - p_{i2} \tag{39}$$

since i is early in the later position. But (39) contradicts (E13).

Hence, this instance is infeasible.

## APPENDIX 3

### Proof of Remark 3

The proof is based on considering various subcases and scenarios that are possible when $j \neq i$, but $i \neq j$. Note that in this case, trailing idle times are such that $S_i^- + p_{j1} - p_{i2} > 0$; $S_j^- + p_{i1} - p_{j2} = 0$. Consistent with the statement of Remark 3, we assume the $P_{i1} \leq P_{j1}$, and show that i precedes j in an optimal solution, and $P_{i2} \leq P_{j2}$.

Considering the following subcases.

I. $\underline{s_i^+ = s_j^+ = 0}$

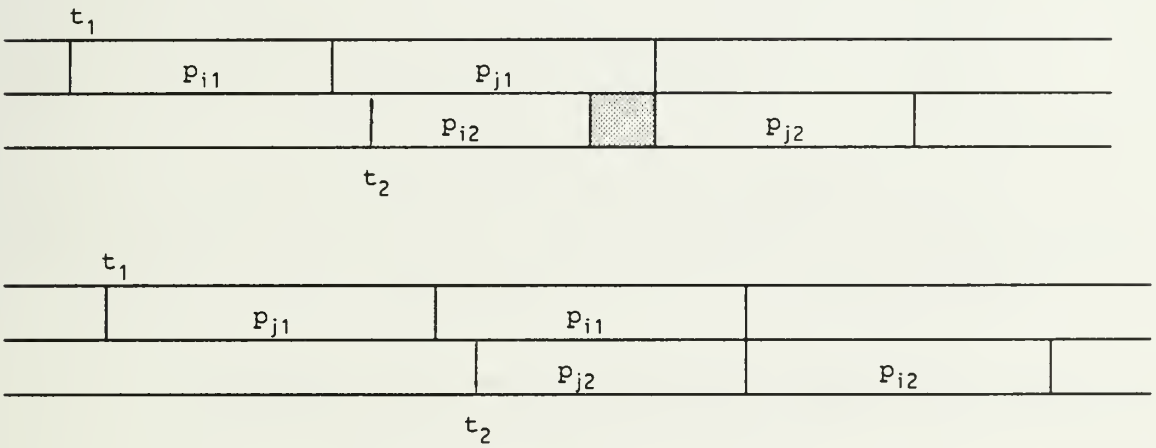


Figure 16: Subcase I

$$\Delta T = [(t_2 + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j)^+]$$
$$- [(t_2 + p_{j2} - d_j)^+ + (t_2 + p_{j2} + p_{i2} - d_i)^+]$$

In this case, in order to show that $P_{i2} \leq P_{j2}$, we need to show that

$$\max\{t_2 + p_{i2}, d_i\} \leq \max\{t_2 + p_{j2}, d_j\}. \qquad (F0)$$

Because $P_{i1} \leq P_{j1}$, we have

$$\max\{t_1+p_{i1}, d_i-p_{i2}\} \leq \max\{t_1+p_{j1}, d_j-p_{j2}\}. \qquad (F1)$$

i) Both i and j late in both positions:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (t_1+p_{i1}+p_{j1}) - (t_2+p_{j2}) < 0.\end{aligned} \qquad (40)$$

In this case, (F0) reduces to

$$t_2 + p_{i2} \leq t_2 + p_{j2}$$

or

$$p_{i2} \leq p_{j2}$$

which is always true because $t_2 + p_{i2} < t_1 + p_{i1} + p_{j1} \leq t_2 + p_{j2}$. Hence, in this case $P_{i2} \leq P_{j2}$.

ii) Both i and j early if scheduled first, late otherwise:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (d_i-d_j) + [(t_1+p_{i1}+p_{j1}) - (t_2+p_{i2})].\end{aligned} \qquad (41)$$

In this case, (F1) reduces to

$$\begin{aligned}d_i - p_{i2} &\leq d_j - p_{j2} \\ \Rightarrow d_i - p_{i2} - d_j &\leq -p_{j2}.\end{aligned} \qquad (F2)$$

From (41) and (F2)

$$\Delta T \leq t_1 + p_{i1} + p_{j1} - (t_2 + p_{j2})$$
$$\leq 0.$$

In this case, (F0) reduces to

$$d_i \leq d_j. \tag{F3}$$

From (F3) and (41), it can be seen that if $\Delta T \leq 0$ then (F3) is true, and the desired result is established.

iii) <u>i early if scheduled first, late otherwise;
j late in either position</u>:

$$\begin{aligned}\Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_2 + p_{j2} - d_j) \\ &\quad - (t_2 + p_{j2} + p_{i2} - d_i) \\ &= [d_i - (t_2 + p_{j2})] + [(t_1 + p_{i1} + p_{j1}) - (t_2 + p_{i2})].\end{aligned} \tag{42}$$

From (F1), in this case we have

$$d_i - p_{i2} \leq \max\{t_1 + p_{j1}, d_j - p_{j2}\}. \tag{F4}$$

If the RHS in (F2) equals $d_j - p_{j2}$, then

$$d_i - p_{i2} \leq d_j - p_{j2} \leq t_2 \tag{42a}$$

because j is late in either position.

From (42) and (42a),

$$\Delta T \leq t_1 + p_{i1} + p_{j1} - (t_2 + p_{j2}) \leq 0.$$

Alternatively if RHS in (F4) equals $t_1 + p_{j1}$, then

$$d_i - p_{i2} \leq t_1 + p_{j1} \tag{42b}$$

and from (42) and (42b)

$$\Delta T \leq -(t_2 + p_{j2}) + (p_{i1} - t_2) < 0.$$

Hence $P_{i1} \leq P_{j1}$ implies $\Delta T \leq 0$.

The second term in the RHS of (42) is non-negative. Hence $\Delta T \leq 0$ implies

$$d_i - (t_2 + p_{j2}) \leq 0. \tag{43}$$

In this case (F0) reduces to

$$d_i \leq t_2 + p_{j2}. \tag{F5}$$

From (43) and (F5), it follows that if $\Delta T \leq 0$, then (F5) is true.

iv) <u>j early if scheduled first, late otherwise; i late in either position</u>:

$$\begin{aligned}\Delta T &= (t_2 + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_2 + p_{j2} + p_{i2} - d_i) \\ &= (t_1 + p_{i1} + p_{j1}) - d_j < 0.\end{aligned}$$

Because j is early in the first position,

$$d_j \geq t_2 + p_{j2} \geq t_1 + p_{i1} + p_{j1}.$$

Hence $\Delta T \leq 0$.

In this case, (F0) reduces to

$$t_2 + p_{i2} \leq d_j$$

which is always true because $d_j \geq t_2 + p_{j2} > t_2 + p_{i2}$. Hence, in this case $\Delta T \leq 0$ and $P_{i2} \leq P_{j2}$.

v) <u>j late in either position; i early in either position</u>:

We need to show that this case is infeasible. (F1) implies in this case

$$d_i - p_{i2} \leq \max\{t_1 + p_{j1}, d_j - p_{j2}\}. \qquad (F6)$$

If the RHS in (F6) equals $t_1 + p_{j1}$, then

$$d_i \leq t_1 + p_{j1} + p_{i2} \leq t_2 + p_{i2}.$$

But this contradicts the fact that i is early in the later position and hence

$$d_i \geq t_2 + p_{j2} + p_{i2}.$$

On the other hand, if the RHS in (F6) is $d_j - p_{j2}$, then

$$\begin{aligned} d_i &\leq d_j - p_{j2} + p_{i2} \\ &< t_2 + p_{i2}, \end{aligned}$$

because j is late in the first position. This contradicts the fact that i is early in the earlier position.

II. $\underline{s_i^+ > 0,\ s_j^+ = 0}$

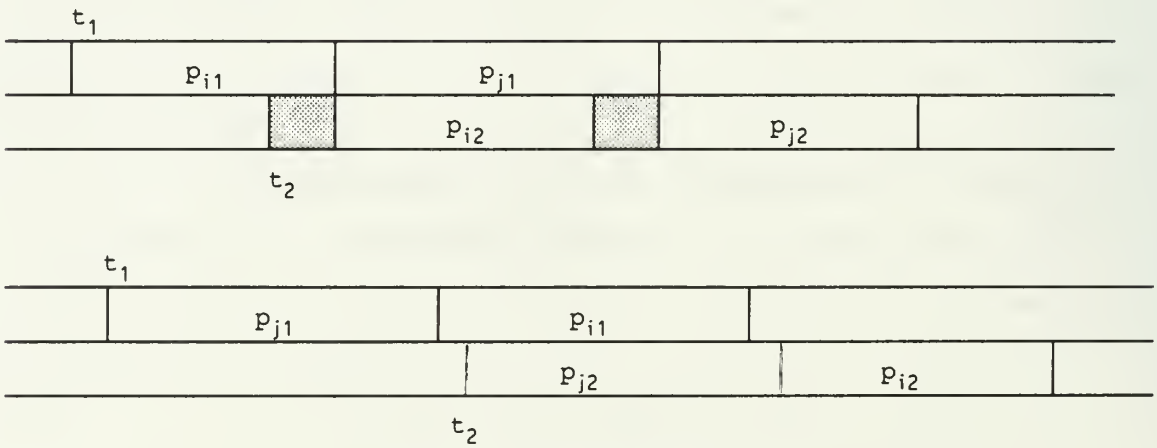


Figure 17: Subcase II

$$\Delta T = [(t_1+p_{i1}+p_{i2}-d_i)^+ + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j)^+]$$
$$- [(t_2+p_{j2}-d_j)^+ + (t_2+p_{j2}+p_{i2}-d_i)^+].$$

In order to show that $P_{i2} \leq P_{j2}$, we need to show that

$$\max\{t_1+p_{i1}+p_{i2}, d_i\} + (t_1+p_{i1}) \leq \max\{t_2+p_{j2}, d_j\} + t_2. \qquad (G0)$$

Consider the following instances:

i) Both i and j are late in either position:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}-d_j) - (t_2+p_{j2}+p_{i2}-d_i) \qquad (44) \\ &= (t_1+p_{i2}) + (t_1+p_{i1}+p_{j1}) - (t_2+p_{j2}) - t_2.\end{aligned}$$

In this case, (G0) reduces to

$$(t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) \leq (t_2+p_{j2}) + t_2. \qquad (G1)$$

From (44) and (G1), it follows that $\Delta T \leq 0$ implies that (C3) is true.

But $p_{i2} < p_{j1}$, hence $t_1 + p_{i2} < t_1 + p_{j1} < t_2$. Also,

$$t_1 + p_{i1} + p_{j1} \leq t_2 + p_{j2}.$$

It follows from (44) that

$$\Delta T < 0.$$

ii) Both i and j are early if scheduled first, late otherwise:

$$\Delta T = (t_1 + p_{i1} + p_{j1}) - (t_2 + p_{i2}) + (d_i - d_j). \tag{45}$$

From (F1), in this case we have

$$d_i - p_{i2} \leq d_j - p_{j2}$$
$$\Rightarrow d_i - d_j - p_{i2} \leq -p_{j2}.$$

Hence,

$$\Delta T \leq (t_1 + p_{i1} + p_{j1}) - (t_2 + p_{j2}) \leq 0. \tag{45a}$$

(G0) reduces to

$$d_i + (t_1 + p_{i1}) \leq d_j + t_2 \tag{G2}$$
$$\text{or} \quad (d_i - d_j) + (t_1 + p_{i1}) - t_2 \leq 0.$$

Note that in this case, $p_{i2} < p_{ji}$. (46)

From (45), (45a), and (46) it follows that (G2) is true.

iii) i early if scheduled early, late otherwise;
j late in either position:

$$\begin{aligned} \Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_2 + p_{j2} - d_j) \\ &\quad - (t_2 + p_{j2} + p_{i2} - d_i) \\ &= (d_i - t_2 - p_{j2}) + (t_1 + p_{i1}) + (p_{j1} - p_{i2}) - t_2. \end{aligned} \tag{47}$$

From (F1), we have in this case

$$d_i - p_{i2} \leq \max\{t_1 + p_{j1}, d_j - p_{j2}\}. \tag{G3}$$

If the RHS in (G3) equals $d_j - p_{j2}$, then

$$d_i - p_{i2} \leq d_j - p_{j2} \leq t_2$$

because j is late in the first position.

But, since $t_2 > t_1 + p_{i1}$, this implies that $d_i \leq t_1 + p_{i1} + p_{i2}$ which contradicts the fact that i is early if scheduled first. Hence RHS in (G3) must equal $t_1 + p_{j1}$. Hence

$$d_i - p_{i2} \leq t_1 + p_{j1} \tag{47a}$$

substituting (47a) into (47) yields

$$\Delta T = (t_1 + p_{i1} + p_{j1}) - (t_2 + p_{j2}) - t_2 + p_{j2} \leq 0. \tag{47b}$$

In this case, (G0) reduces to

$$d_i + (t_1 + p_{i1}) \leq (t_2 + p_{j2}) + t_2. \tag{G4}$$

From (46), (47a), and (47b) it follows that (G4) is true.

iv) j early if scheduled first, late otherwise;
i late in either position:

$$\begin{aligned}\Delta T &= (t_1+p_{i1}+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_2+p_{j2}+p_{i2}-d_i) \\ &= (t_1+p_{i1}+p_{j1}) + (t_1+p_{i1}) - d_j - t_2.\end{aligned} \tag{48}$$

From (F1), we have

$$t_1 + p_{i1} \leq d_j - p_{j2}.$$

Substituting this expression in (48), we have

$$\Delta T \leq (t_1+p_{i1}+p_{j1}) - (t_2+p_{j2}) \leq 0. \tag{48a}$$

In this case (G0) reduces to

$$(t_1+p_{i1}+p_{i2}) + (t_1+p_{i1}) \leq d_j + t_2. \tag{G5}$$

From (48) and (48a), it follows that (G5) is true.

v) j late in either position; i early in either position:

The proof of infeasibility of this case is identical to that of Subcase IV; it is, therefore, omitted.

III. $s_i^+ = 0,\ s_j^+ > 0$

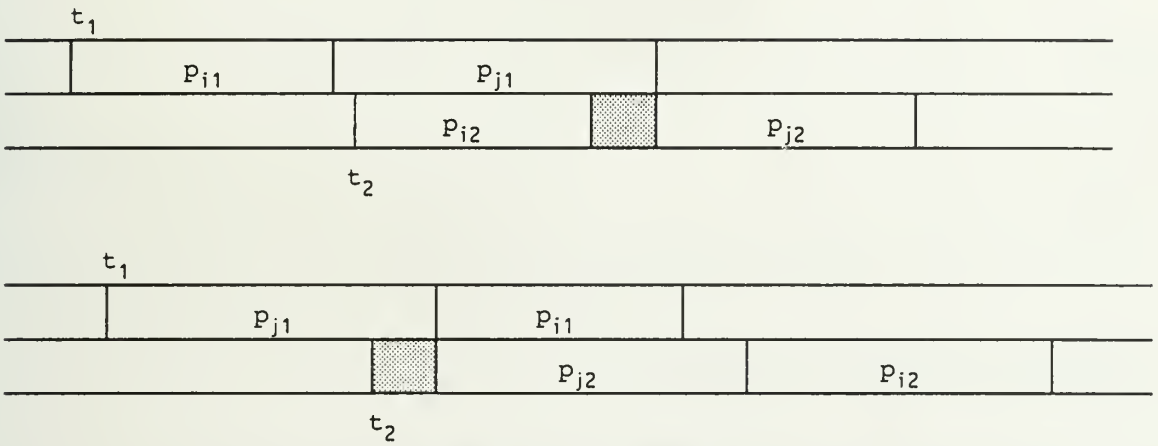


Figure 18: Subcase III

$$\Delta T = (t_2+p_{i2}-d_i)^+ + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j)^+ - (t_1+p_{j1}+p_{j2}-d_j)^+ - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i)^+.$$

(F0) reduces to

$$\max\{t_2+p_{i2}, d_i\} + t_2 \le \max\{t_1+p_{j1}+p_{j2}, d_j\} + (t_1+p_{j1}). \qquad (H0)$$

Consider the following instances:

i) Both i and j late in either position:

$$\begin{aligned}\Delta T &= (t_2+p_{i2}-d_i) + (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) \\ &\quad - (t_1+p_{j1}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= t_2 - (t_1+p_{j1}+p_{j2}) < 0.\end{aligned} \qquad (49)$$

(H0) reduces to

$$(t_2+p_{i2}) + t_2 \leq (t_1+p_{j1}+p_{j2}) + (t_1+p_{j1}). \qquad (H1)$$

But

$$t_2 \leq t_1 + p_{j1} \qquad (50)$$

and

$$t_2 + p_{i2} < t_1 + p_{i1} + p_{j1} \leq t_1 + p_{j1} + p_{j2}.$$

Hence (H8) is always satisfied.

ii) Both i and j early if scheduled first, late otherwise

$$\begin{aligned} \Delta T &= (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= (d_i-d_j) + (p_{i1}-p_{i2}) \\ &= (d_i-d_j) + (t_1+p_{i1}+p_{j1}) - (t_1+p_{j1}+p_{i2}) \qquad (51) \\ &= (d_i-d_j) + \{(t_1+p_{i1}+p_{j1}) - p_{i2}\} - (t_1+p_{j1}) \\ &> (d_i-d_j) + t_2 - (t_1+p_{j1}). \end{aligned}$$

From (F1), we have

$$\max\{t_1+p_{i1}, d_i-p_{i2}\} \leq d_j - p_{j2}.$$

Hence

$$d_i - d_j \leq p_{i1} - p_{j2}. \qquad (51a)$$

Hence

$$\begin{aligned}\Delta T &= (d_i - d_j) + (p_{i1} - p_{i2}) \\ &\leq d_i - d_j + p_{j2} - p_{i2}, \text{ because } p_{i1} < p_{j2} \\ &\leq 0 \quad \text{from (51a).}\end{aligned} \tag{51b}$$

(H0) reduces to

$$d_i + t_2 \leq d_j + (t_1 + p_{j1})$$

$$\text{or} \quad (d_i - d_j) + t_2 - (t_1 + p_{j1}) \leq 0. \tag{H2}$$

From (51a) and (51b), it follows that (H2) is true.

iii) <u>i early if scheduled first, late otherwise;
j late in either position</u>:

$$\begin{aligned}\Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i) \\ &= d_i - (t_1 + p_{j1} + p_{j2}) + (p_{i1} - p_{i2}).\end{aligned} \tag{52}$$

From (F1), we have

$$d_i - p_{i2} \leq t_1 + p_{j1}.$$

Hence,

$$\Delta T \leqq p_{i1} - p_{j2} \leq 0.$$

(H0) reduces to

$$d_i + t_2 \leq (t_1 + p_{j1} + p_{j2}) + (t_1 + p_{j1}). \qquad (H3)$$

By carrying out algebraic manipulations in (52) similar to ii) above, it can shown that (H3) is true in this scenario as well.

iv) <u>j early if scheduled first, late otherwise; i late in either position</u>:

$$\begin{aligned}\Delta T &= (t_2 + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i) \\ &= t_2 + p_{i1} - d_j < t_1 + p_{j1} + p_{i1} - d_j < 0\end{aligned}$$

since $d_j$ is early if scheduled first.

(H0) reduces to

$$t_2 + p_{i2} + t_2 \leq d_j + (t_1 + p_{j1})$$

which is always true from (50) and the fact that

$$d_j \geq t_1 + p_{j1} + p_{j2} > t_2 + p_{i2}.$$

Hence, in this instance, $P_{i2} \leq P_{j2}$ and $\Delta T \leq 0$ always.

v) <u>j late in either position; i early in either position</u>:

Need to show that this case is infeasible. From (F1), in this case we have

$$\begin{aligned}\max\{t_1 + p_{i1}, d_i - p_{i2}\} &\leq t_1 + p_{j1} \\ \Rightarrow d_i &\leq t_1 + p_{j1} + p_{i2}\end{aligned}$$

which contradicts the fact that i is early in the later position.

IV. $\underline{s_i^+ > 0;\ s_j^+ > 0}$

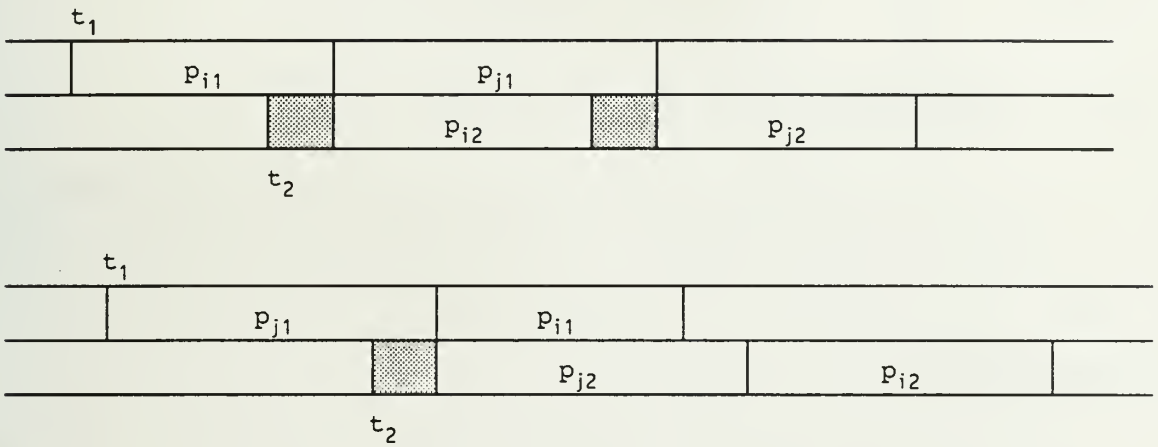


Figure 19: Subcase IV

$$\Delta T = (t_1 + p_{i1} + p_{i2} - d_i)^+ + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j)^+ - (t_1 + p_{j1} + p_{j2} - d_j)^+ - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i)^+.$$

(F0) reduces to

$$\max\{t_1 + p_{i1} + p_{i2}, d_i\} + (t_1 + p_{i1}) \leq \max\{t_1 + p_{j1} + p_{j2}, d_j\} + (t_1 + p_{j1}). \qquad (I0)$$

Consider the following instances:

i) Both i and j late in both positions:

$$\begin{aligned}\Delta T &= (t_1 + p_{i1} + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i) \\ &= 2p_{i1} - (p_{j1} + p_{j2}).\end{aligned} \qquad (53)$$

From (F1), we have

$$t_1 + p_{i1} \leq t_1 + p_{j1}$$

$$\text{or} \qquad p_{i1} \leq p_{j1}.$$

But, as seen from Figure 19, $p_{i1} \leq p_{j2}$ as well. Hence,

$$2p_{i1} \leq p_{j1} + p_{j2} \quad \text{and} \quad \Delta T \leq 0. \tag{53a}$$

(I0) reduces to

$$(t_1+p_{i2}+p_{i2}) + (t_1+p_{i1}) \leq (t_1+p_{j1}+p_{j2}) + (t_1+p_{j1})$$

$$\text{or} \quad 2p_{i1} + p_{i2} \leq (p_{j1}+p_{j2}) + p_{j1}. \tag{I1}$$

But $p_{i2} < p_{j1}$, because

$$s_i^- + p_{j1} - p_{i2} > 0. \tag{54}$$

Hence, from (53) and (53a) (I1) is true.

ii) <u>Both i and j early if scheduled first, late otherwise</u>:

$$\begin{aligned} \Delta T &= (t_1+p_{i1}+p_{j1}+p_{j2}-d_j) - (t_1+p_{j1}+p_{j2}+p_{i2}-d_i) \\ &= (d_i-d_j) + (p_{i1}-p_{i2}). \end{aligned} \tag{55}$$

From (F1), we have in this case

$$
\begin{aligned}
& d_i - p_{i2} \le d_j - p_{j2} \\
& \quad \Rightarrow d_i - d_j + (p_{j2} - p_{i2}) \le 0 \\
& \quad \Rightarrow d_i - d_j + (p_{i1} - p_{i2}) \le 0 \quad \text{because } p_{i1} \le p_{j2} \\
& \quad \Rightarrow \Delta T \le 0 \quad \text{from (55)}.
\end{aligned}
\tag{55a}
$$

(I0) reduces to

$$d_i + p_{i1} \le d_j + p_{i2}. \tag{I2}$$

From (55a), it follows that (I2) is true.

iii) <u>i early if scheduled first, late otherwise;
j late in either position</u>:

$$
\begin{aligned}
\Delta T &= (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) - (t_1 + p_{j1} + p_{j2} - d_j) \\
&\quad - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i) \\
&= d_i - (t_1 + p_{j1} + p_{j2}) + (p_{i1} - p_{i2}).
\end{aligned}
\tag{56}
$$

In this case, (F1) reduces to

$$
\begin{aligned}
& d_i - p_{i2} \le t_1 + p_{j1} \\
\text{or } & d_i - (t_1 + p_{j1} - p_{i2}) \le 0 \\
\text{or } & d_i - (t_1 + p_{j1} - p_{i2}) + (p_{i1} - p_{j2}) \le 0, \quad \text{because } p_{i1} \le p_{j2} \\
\Rightarrow \quad & \Delta T \le 0, \quad \text{from (56)}.
\end{aligned}
\tag{56a}
$$

(I0) reduces to

$$d_i + (t_1 + p_{i1}) \leq (t_1 + p_{j1} + p_{j2}) + (t_1 + p_{j1})$$

$$\text{or} \quad d_i - (t_1 + p_{j1} + p_{j2}) + (p_{i1} - p_{j1}) \leq 0. \tag{I3}$$

From (56) and (56a), it follows that (I3) is true.

iv) <u>j early if scheduled first, late otherwise; i late in either position</u>:

$$\begin{aligned} \Delta T &= (t_1 + p_{i1} + p_{i2} - d_i) + (t_1 + p_{i1} + p_{j1} + p_{j2} - d_j) \\ &\quad - (t_1 + p_{j1} + p_{j2} + p_{i2} - d_i) \\ &= p_{i1} + (t_1 + p_{i1}) - d_j. \end{aligned} \tag{57}$$

In this case, (F1) reduces to

$$\begin{aligned} & t_1 + p_{i1} \leq d_j - p_{j2} \\ \text{or} \quad & t_1 + p_{i1} - d_j + p_{j2} \leq 0 \\ \Rightarrow \quad & t_1 + p_{i1} - d_j + p_{i1} \leq 0 \quad \text{because } p_{i1} \leq p_{u2} \\ \Rightarrow \quad & \Delta T \leq 0, \quad \text{from (57)}. \end{aligned} \tag{57a}$$

(I0) reduces to

$$(t_1 + p_{i1} + p_{i2}) + (t_1 + p_{i1}) \leq d_j + (t_1 + p_{j1})$$

$$\text{or} \quad (t_1 + p_{i1}) - d_j + (p_{i1} + p_{i2} - p_{j1}) \leq 0. \tag{I4}$$

From (57) and (57a), it follows that (I4) is true.

v) <u>j late in either position; i early in either position</u>:

We need to show that this case is infeasible. (F1) reduces to

$$d_i - p_{i2} \leq t_1 + p_{j1}$$

$$\Rightarrow \quad d_i \leq t_1 + p_{j1} + p_{i2}$$

which contradicts the fact that i is early in the later position.

This completes the proof of the first part of Remark 3. The proof for the second part of this remark is similar, and it is consequently omitted.

HECKMAN
BINDERY INC.
JUN 95
Bound-To-Please® N. MANCHESTER, INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA
3 0112 060295836